IN-62
247 p

NASA Contractor Report 191465

Towards the Formal Specification of the Requirements and Design of a Processor Interface Unit — HOL Listings

David A. Fura
The Boeing Company
Seattle, Washington

Phillip J. Windley
University of Idaho
Moscow, Idaho

Gerald C. Cohen
The Boeing Company
Seattle, Washington

NASA Contract NAS1-18586
November 1993

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23681-0001

(NASA-CR-191465) TOWARDS THE FORMAL SPECIFICATION OF THE REQUIREMENTS AND DESIGN OF A PROCESSOR INTERFACE UNIT: HOL LISTINGS (Boeing Defense and Space Group) 247 p

N94-23252

Unclas

G3/62 0198597

Preface

This document was generated in support of NASA contract NAS1-18586, Design and Validation of Digital Flight Control Systems Suitable for Fly-By-Wire Applications, Task Assignment 10. Task 10 is concerned with the formal specification and verification of a processor interface unit.

This report contains the HOL listings of the formal specification of the design and partial requirements for a processor interface unit using the HOL theorem-proving system. The specification approach is described in NASA CR-4521. The processor interface unit is a single-chip subsystem within a fault-tolerant embedded system under development within the Boeing Defense & Space Group. It provides the opportunity to investigate the specification and verification of a real-world subsystem within a commercially-developed fault-tolerant computer.

The NASA technical monitor for this work is Sally Johnson of the NASA Langley Research Center, Hampton, Virginia.

The work was accomplished at the Boeing Company, Seattle, Washington and the University of Idaho, Moscow, Idaho. Personnel responsible for the work include:

Boeing Defense & Space Group:
D. Gangsaas, Responsible Manager
T. M. Richardson, Program Manager

Boeing Defense & Space Group:
Gerald C. Cohen, Principal Investigator
David A. Fura, Researcher

University of Idaho:
Dr. Phillip J. Windley, Chief Researcher

PRECEDING PAGE BLANK NOT FILMED

Contents

1 Introduction 1
2 Supporting Theories 2
3 PIU Design Specification 39
3.1 PIU-Applicable Definitions 39
3.2 P-Port Definitions 40
3.3 M-Port Definitions 60
3.4 R-Port Definitions 83
3.5 C-Port Definitions 120
3.6 SU-Cont Definitions 166
4 PIU Requirements Specification 186
4.1 PIU Transaction-Level Specification 186
4.2 P-Port Transaction-Level Specification 202
4.3 M-Port Transaction-Level Specification 215
4.4 C-Port Transaction-Level Specification 227
4.5 R-Port Transaction-Level Specification 232

PAGE IV INTENTIONALLY BLANK

PRECEDING PAGE NOT FILMED

1 Introduction

This technical report contains the HOL listings of the specification of the design and major portions of the requirements for a commercially-developed processor interface unit (or PIU). The PIU is an interface chip performing memory-interface, bus-interface, and additional support services for a commercial microprocessor within a fault-tolerant computer system. This system, the Fault-Tolerant Embedded Processor (FTEP), is targeted towards applications in avionics and space requiring extremely high levels of mission reliability, extended maintenance-free operation, or both.

This report contains the actual HOL listings of the PIU specification as it currently exists. For those interested in an informal description of the PIU specification, NASA CR-4521 contains a discussion of the modeling issues involved in the PIU specification, as well as an overview of the specification itself.

Section 2 of this report contains general-purpose HOL theories that support the PIU specification. These theories include definitions for the hardware components used in the PIU, our implementation of n-bit words, and our implementation of temporal logic.

Section 3 contains the HOL listings for the PIU *design* specification. Aside from the PIU internal bus (I_Bus), this specification is complete.

Section 4 contains the HOL listings for a major portion of the PIU *requirements* specification. Specifically, it contains most of the definition for the PIU behavior associated with memory accesses initiated by the local processor.

2 Supporting Theories

This section contains general-purpose theories used in the PIU specification. The theories *array_def*, *wordn_def*, *busn_def*, and *templogic_def* contain our implementations for arrays, n-bit words (of bit-type bool), 4-valued logic, and temporal logic, respectively. The theories *gates_def1*, *latches_def*, *ffs_def*, *counters_def*, *datapaths_def*, and *buses_def* contain component models for logic-gates, latches, flip-flops, counters, datapath elements, and bus nodes, respectively.

```
%-------------------------------------------------------------

   File:          array_def.ml

   Author:        (c) P. J. Windley 1992

   Description:

   Prove auxilliary theorems about functions so that functions
   can be easily used to represent arrays.

   Modification History:

   24FEB92 -- Original file.  Many of the theorems included were
              motivated by theorems defined on lists in
              list_aux.ml.
   26FEB92 -- [DAF] Modified order of parameters in calls to
   ALTER, MALTER, SUBARRAY to match simulation
   language syntax. Added definition of ELEMENT.
   04OCT92 -- [DAF] Added theorem SUBARRAY_MALTER_IDENT.
   14OCT92 -- [DAF] Added definition DEF_SIZE.
   29OCT92 -- [DAF] Added definition ARBN (from wordn_def).
   14DEC92 -- [DAF] Added theorem SUB_SUBARRAY.

-------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

system 'rm array_def.th';;

new_theory 'array_def';;

loadf 'aux_defs';;

load_library 'reduce';;

%-------------------------------------------------------------
 Auxilliary array definitions and theorems.

 We will use functions to represent arrays.  The definition
 that follows defines a ALTER function that can be used to set
 the nth member of an array.  The following lemmas are useful
 in reasoning about array operations.
-------------------------------------------------------------%

let ALTER_DEF = new_definition
   ('ALTER_DEF',
    "ALTER (f:*->**) n x = (\m. (m = n) => x | (f m))"
   );;

let ALTER_THM = prove_thm
   ('ALTER_THM',
    "ALTER (f:*->**) n x y = (y = n) => x | (f y)",
    REWRITE_TAC [ALTER_DEF]
    THEN BETA_TAC
    THEN REFL_TAC
   );;
```

```
%-------------------------------------------------------------------
 ALTER_EQUAL is simlar to the EL_SET_EL lemma for lists.
-------------------------------------------------------------------%
let ALTER_EQUAL = prove_thm
   ('ALTER_EQUAL',
    "! x n (f:*->**) . (ALTER f n x) n = x",
    REPEAT GEN_TAC
    THEN REWRITE_TAC [ALTER_DEF]
    THEN BETA_TAC
    THEN REWRITE_TAC []
   );;

%-------------------------------------------------------------------
 ALTER_NON_EQUAL is similar to NOT_EL_SET_EL for lists.
-------------------------------------------------------------------%
let ALTER_NON_EQUAL = prove_thm
   ('ALTER_NON_EQUAL',
    "! n m (f:*->**) x .
     ~(n = m) ==>
     (f n = (ALTER f m x) n)",
    REPEAT GEN_TAC
    THEN REWRITE_TAC [ALTER_THM]
    THEN STRIP_TAC
    THEN ASM_REWRITE_TAC []
   );;

%-------------------------------------------------------------------
 ALTER_COMMUTES is similar to SET_EL_SET_EL for lists.
-------------------------------------------------------------------%
let ALTER_COMMUTE = prove_thm
   ('ALTER_COMMUTE',
    "! (d1:*) d2 (f:*->**) (x:**) y .
     ~(d1 = d2) ==>
     ((ALTER (ALTER f d2 x) d1 y) =
      (ALTER (ALTER f d1 y) d2 x))",
    REPEAT GEN_TAC
    THEN CONV_TAC (ONCE_DEPTH_CONV FUN_EQ_CONV)
    THEN REWRITE_TAC [ALTER_THM]
    THEN STRIP_TAC
    THEN GEN_TAC
    THEN REPEAT COND_CASES_TAC
    THEN ASM_REWRITE_TAC []
    THEN UNDISCH_TAC "~((d1:*) = d2)"
    THEN ASSUM_LIST (\thl . REWRITE_TAC (map SYM_RULE thl))
   );;

%-------------------------------------------------------------------

 Until now, it hasn't mattered what the type of the subscript is
 and so the previous lemmas were all general, even though
 someone using them to representa arrays, would probably be
 using numbers as subscripts.

 Now, we want to reason about subarrays given as a sequence from
 a starting value to an ending value.  This presupposes that the
 subscripts can be totally ordered.  To make life easy, we won't
 be that general, but will use numbers as subscripts.

-------------------------------------------------------------------%

let SUBARRAY_DEF = new_definition
   ('SUBARRAY_DEF',
    "! n m (f:num->*) .
     SUBARRAY f (m,n) = \x. ((x+n) <= m) => f(x+n) | ARB"
   );;

let SUBARRAY_THM = prove_thm
   ('SUBARRAY_THM',
    "! n m (f:num->*) .
     SUBARRAY f (m,n) x = ((x+n) <= m) => f(x+n) | ARB",
    REPEAT GEN_TAC
    THEN REWRITE_TAC [SUBARRAY_DEF]
```

```
    THEN BETA_TAC
    THEN REFL_TAC
   );;

let ELEMENT_DEF = new_definition
('ELEMENT_DEF',
 "! m (f:num->*) .
  ELEMENT f (m) = f m"
);;

%-------------------------------------------------------------
 MALTER alters multiple values in an array.
-------------------------------------------------------------%

let MALTER_DEF = new_definition
   ('MALTER_DEF',
    "! n m f (g:num->*) .
     MALTER f (m,n) g =
      \x. (n <= x /\ x <= m) => g (x-n) | f x"
   );;

let MALTER_THM = prove_thm
   ('MALTER_THM',
    "! n m (x:num) g (f:num->*) .
     MALTER f (m,n) g x = (n <= x /\ x <= m) => g (x-n) | f x",
    REPEAT GEN_TAC
    THEN REWRITE_TAC [MALTER_DEF]
    THEN BETA_TAC
    THEN REFL_TAC
   );;


let MALTER_SUBARRAY_IDENT = prove_thm
   ('MALTER_SUBARRAY_IDENT',
    "!n m (f:num->*) . MALTER f (m,n) (SUBARRAY f (m,n)) = f",
    REPEAT GEN_TAC
    THEN CONV_TAC (ONCE_DEPTH_CONV FUN_EQ_CONV)
    THEN REWRITE_TAC [MALTER_THM;SUBARRAY_THM]
    THEN GEN_TAC
    THEN REPEAT COND_CASES_TAC
    THEN ASM_REWRITE_TAC []
    THEN ASSUM_LIST (\thl . MAP_EVERY ASSUME_TAC
       (flat (map CONJUNCTS (filter (is_conj o concl) thl))))
    THEN IMP_RES_TAC SUB_ADD
    THEN TRY (UNDISCH_TAC "~((n' - n) + n) <= m")
    THEN ASM_REWRITE_TAC []
   );;

let MALTER_SUBARRAY_SUBSCRIPTS = prove_thm
   ('MALTER_SUBARRAY_SUBSCRIPT',
    "!n m x (f:num->*) g .
     MALTER f (m,n) (SUBARRAY g (m,n)) x =
            (n <= x /\ x <= m) => g x | f x",
    REPEAT GEN_TAC
    THEN CONV_TAC (ONCE_DEPTH_CONV FUN_EQ_CONV)
    THEN REWRITE_TAC [MALTER_THM;SUBARRAY_THM]
    THEN REPEAT COND_CASES_TAC
    THEN ASM_REWRITE_TAC []
    THEN ASSUM_LIST (\thl . MAP_EVERY ASSUME_TAC
       (flat (map CONJUNCTS (filter (is_conj o concl) thl))))
    THEN IMP_RES_TAC SUB_ADD
    THEN TRY (UNDISCH_TAC "~((x - n) + n) <= m")
    THEN ASM_REWRITE_TAC []
   );;

let lemma1 = TAC_PROOF
   (([], "!(a :bool) (b c d :*) .  a => (a => b | c) | d = (a => b | d)"),
    REPEAT GEN_TAC
    THEN REPEAT COND_CASES_TAC
    THEN REWRITE_TAC[]
   );;
```

```
let lemma2 = TAC_PROOF
   (([], "! (a b c :num) . (b <= c) ==> (((a + b) <= c) = (a <= c - b))"),
    REPEAT STRIP_TAC
    THEN PURE_REWRITE_TAC [SPECL ["a:num";"c:num - b:num";"b:num"]
                           (SYM_RULE LESS_EQ_MONO_ADD_EQ)]
    THEN IMP_RES_TAC (SPECL ["b:num";"c:num"] SUB_ADD)
    THEN ASM_REWRITE_TAC[]
   );;

let ARBN = new_definition
   ('ARBN',
    "(ARBN:num->*) = \n. ARB"
   );;

let SUBARRAY_MALTER_IDENT = prove_thm
   ('SUBARRAY_MALTER_IDENT',
    "!(m n :num) (f g h :num->*) .
     ((n <= m) /\
      (g = MALTER ARBN ((m-n),0) h))
       ==>
      (SUBARRAY (MALTER f (m,n) g) (m,n) = g)",
    REPEAT STRIP_TAC
    THEN CONV_TAC (ONCE_DEPTH_CONV FUN_EQ_CONV)
    THEN ASM_REWRITE_TAC [MALTER_THM;SUBARRAY_THM;ARBN;
                          SPECL ["n':num";"n:num"] ADD_SUB;
                          SPEC "n':num" ZERO_LESS_EQ;
                          SPEC "n':num" SUB_0;
                          SPECL ["n:num";"n':num"]
                                (ONCE_REWRITE_RULE [ADD_SYM] LESS_EQ_ADD)]
    THEN IMP_RES_TAC (SPECL ["n':num";"n:num";"m:num"] (SYM_RULE lemma2))
    THEN ASM_REWRITE_TAC[lemma1]
   );;

let SUB_SUBARRAY = prove_thm
   ('SUB_SUBARRAY',
    "! (f :num->*) (m n p :num) .
     (m <= p) ==>
        (SUBARRAY (SUBARRAY f (p,0)) (m,n) = SUBARRAY f (m,n))",
    REPEAT STRIP_TAC
    THEN CONV_TAC (ONCE_DEPTH_CONV FUN_EQ_CONV)
    THEN REWRITE_TAC [SUBARRAY_THM;ADD_CLAUSES]
    THEN GEN_TAC
    THEN ASM_CASES_TAC "(n'+n) <= m"
    THEN IMP_RES_TAC (SPECL ["n'+n";"m:num";"p:num"] LESS_EQ_TRANS)
    THEN ASM_REWRITE_TAC[]
   );;

let DEF_SIZE = new_definition
   ('DEF_SIZE',
    "! (f :num->*) (n :num) .
     DEF_SIZE f n = MALTER ARBN (n,0) f"
   );;

close_theory();;


%-------------------------------------------------------------------

   File:          wordn_def.ml

   Description:

   Defines a theory of words which contains a definition for
   converting between functions from numbers to booleans and
   natural numbers and proves various useful theorems about
   this definition.  This file is based on a theory that was
   orginally authored by Graham Birtwhistle of the University
   of Calgary in 1988.

   Authors: (c) Graham Birtwhistle, Phillip Windley, 1988, 1992

   Modification History:
```

```
   28FEB92 -- [PJW] Original file from words.ml

   10MAR92 -- [PJW] Added definition of WORDN.
   13MAR92 -- [DAF] Added definitions of bv, SETN, RSTN, GNDN,
   NOTN, INCN, DECN.

   13OCT92 -- [DAF] Added definition of ANDN, ORN.

   01DEC92 -- [DAF] Added theorems VAL_WORDN_IDENT_3,
                                   WORDN_3_NOT_EQUAL.

------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/']);;

system '/bin/rm wordn_def.th';;

new_theory 'wordn_def';;

loadf 'aux_defs';;

map new_parent ['piuaux_def'];;

map load_parent ['array_def';'ineq'];;

new_type_abbrev ('wordn',":num->bool");;

load_library 'reduce';;

%------------------------------------------------------------------
 Definitions
------------------------------------------------------------------%

let bv = new_definition
('bv',
 "! (b:bool) .
  bv b = (b) => 1 | 0"
);;

let VAL = new_prim_rec_definition
  ('VAL',
   "(VAL 0 (f:wordn) = bv (f 0))
     /\
    (VAL (SUC n) f = ((2 EXP (SUC n)) * (bv (f (SUC n)))) + VAL n f)"
  );;

let pos_val = new_definition
   ('pos_val',
    "! (x:wordn) (y:num) .
          pos_val x y = (bv(x y)) * (2 EXP y)"
   );;

let ONES = new_prim_rec_definition
  ('ONES',
   "(ONES 0 a = (a 0))
     /\
    (ONES (SUC n) a = (a(SUC n)) /\ (ONES n a))
   ");;

let ZEROS = new_prim_rec_definition
  ('ZEROS',
   "(ZEROS 0 a = ~(a 0))
     /\
    (ZEROS (SUC n) a = ~(a(SUC n)) /\ (ZEROS n a))
   ");;

let WORDN = new_definition
    ('WORDN',
     "! (n x:num) .
      WORDN n x = \m. (m <= n) => ((x DIV (2 EXP m)) MOD 2 = 1) | ARB"
```

```
    );;

let SETN = new_definition
('SETN',
 "! (x:num) . SETN x = \(n:num). (n <= x) => T | ARB"
);;

let RSTN = new_definition
('RSTN',
 "! (x:num) . RSTN x = \(n:num). (n <= x) => F | ARB"
);;

let GNDN = new_definition
('GNDN',
 "! (x:num) (t:time) .
     GNDN x t = ((\(n:num). (n <= x) => F | ARB),
                 (\(n:num). (n <= x) => F | ARB))"
);;

let NOTN = new_definition
('NOTN',
 "! (x:num) (f:wordn) . NOTN x f = \(n:num) . (n <= x) => ~(f n) | ARB"
);;

let ANDN = new_definition
   ('ANDN',
    "! (x:num) (f g :wordn) .
     ANDN x f g = \(n:num) . (n <= x) => ((f n) /\ (g n)) | ARB"
   );;

let ORN = new_definition
   ('ORN',
    "! (x:num) (f g :wordn) .
     ORN x f g = \(n:num) . (n <= x) => ((f n) \/ (g n)) | ARB"
   );;

let INCN = new_definition
('INCN',
 "! n f .
  INCN n f = (ONES n f) => RSTN n | WORDN n ((VAL n f) + 1)"
);;

let DECN = new_definition
('DECN',
 "! n f .
  DECN n f = (ZEROS n f) => SETN n | WORDN n ((VAL n f) - 1)"
);;

let VAL_WORDN_IDENT_1 = prove_thm
   ('VAL_WORDN_IDENT_1',
    "! n :num . n <= 3 ==> (VAL 1 (WORDN 1 n) = n)",
    REWRITE_TAC [VAL;WORDN;bv;num_CONV "1";
                 LESS_EQ_3_CASES]
    THEN BETA_TAC
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
    THEN REDUCE_TAC
   );;

let SIZE_1 = new_definition
   ('SIZE_1',
    "! x :wordn . SIZE_1 x = !n:num. ~n <= 1 ==> (x n = ARB)"
   );;

let WORDN_VAL_IDENT_1 = prove_thm
   ('WORDN_VAL_IDENT_1',
    "! x :wordn . SIZE_1 x ==> (WORDN 1 (VAL 1 x) = x)",
    REWRITE_TAC [SIZE_1;num_CONV "1";VAL;WORDN;bv]
    THEN GEN_TAC
    THEN REDUCE_TAC
    THEN REPEAT STRIP_TAC
    THEN CONV_TAC (ONCE_DEPTH_CONV FUN_EQ_CONV)
```

```
    THEN BETA_TAC
    THEN GEN_TAC
    THEN COND_CASES_TAC
    THEN RES_TAC
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC LESS_EQ_1_CASES
    THEN ASM_REWRITE_TAC[]
    THEN BOOL_CASES_TAC "(x 0):bool"
    THEN BOOL_CASES_TAC "(x 1):bool"
    THEN ASM_REWRITE_TAC[]
    THEN REDUCE_TAC
   );;

let SIZE_SUBARRAY_1 = prove_thm
   ('SIZE_SUBARRAY_1',
    "! x :wordn .  SIZE_1 (SUBARRAY x (1,0))",
    REWRITE_TAC [SIZE_1;SUBARRAY_DEF;ADD_CLAUSES]
    THEN BETA_TAC
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
   );;

let VAL_WORDN_IDENT_3 = prove_thm
   ('VAL_WORDN_IDENT_3',
    "! n :num . n <= 15 ==> (VAL 3 (WORDN 3 n) = n)",
    REWRITE_TAC [VAL;WORDN;bv;num_CONV "3";num_CONV "2";num_CONV "1";
                 LESS_EQ_15_CASES]
    THEN BETA_TAC
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
    THEN REDUCE_TAC
   );;

let lemma1 = TAC_PROOF
   (([], "! (x y :*) (f :*->**) . (x = y) ==> (f x = f y)"),
    REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
   );;

% |- !n m. n <= 3 ==> m <= 3 ==> ~(m = n) ==> ~(WORDN m = WORDN n) %

let WORDN_1_NOT_EQUAL = save_thm
   ('WORDN_1_NOT_EQUAL',
    GEN_ALL
     (DISCH_ALL
       (REWRITE_RULE
         [MP
           (SPEC "n:num" VAL_WORDN_IDENT_1)
           (ASSUME "n <= 3")
         ;
         MP
           (SPEC "m:num" VAL_WORDN_IDENT_1)
           (ASSUME "m <= 3")]
         (CONTRAPOS (ISPECL ["WORDN 1 m";"WORDN 1 n";"VAL 1"] lemma1))))
   );;

% |- !n m. n <= 15 ==> m <= 15 ==> ~(m = n) ==> ~(WORDN m = WORDN n) %

let WORDN_3_NOT_EQUAL = save_thm
   ('WORDN_3_NOT_EQUAL',
    GEN_ALL
     (DISCH_ALL
       (REWRITE_RULE
         [MP
           (SPEC "n:num" VAL_WORDN_IDENT_3)
           (ASSUME "n <= 15")
         ;
         MP
           (SPEC "m:num" VAL_WORDN_IDENT_3)
           (ASSUME "m <= 15")]
         (CONTRAPOS (ISPECL ["WORDN 3 m";"WORDN 3 n";"VAL 3"] lemma1))))
   );;
```

```
%-------------------------------------------------------------
 Theorems
-------------------------------------------------------------%

let VAL_WORDN_IDENT = mk_thm
   ([],"! (n x :num) .  (x < (2 EXP (SUC n))) ==> (VAL n (WORDN n x) = x)");;

let WORDN_VAL_IDENT = mk_thm
   ([],
    "! (n :num)(x :wordn) .
     (VAL n x < (2 EXP (SUC n))) ==> (WORDN n (VAL n x) = x)"
   );;

% Removed theorems for now 13MAR92. [DAP]

let MAXWORD = prove_thm
   ('MAXWORD',
    " ! n b. (VAL n b) < (2 EXP (SUC n))",
    INDUCT_TAC
    THEN GEN_TAC
    THEN PURE_ONCE_REWRITE_TAC [ VAL ]
    THENL [
        PURE_REWRITE_TAC [ EXP; MULT_CLAUSES; maxbit ]
      ;
BOOL_CASES_TAC "(b(SUC n)):bool"
THEN REWRITE_TAC [ bv; ADD_CLAUSES; MULT_CLAUSES ]
THENL [
    PURE_ASM_REWRITE_TAC
      [ SPEC "SUC n" EXP_DOUBLES ; MULT_BY_2 ;
(PURE_ONCE_REWRITE_RULE [ADD_SYM] LESS_MONO_ADD_EQ) ]
  ;
    POP_ASSUM
      (\ th. ACCEPT_TAC
(MATCH_MP LESS_TRANS
  (CONJ (SPEC "b" th)
     (SPEC "SUC n" EXP_MONO))))
  ]
      ]
  );;

let MAXWORD2 =  prove_thm
   ('MAXWORD2',
    "! n a cin . ((VAL n a) + (bv cin)) <= (2 EXP (SUC n))",
     REPEAT GEN_TAC
     THEN PURE_ONCE_REWRITE_TAC [bv]
     THEN COND_CASES_TAC
     THEN PURE_REWRITE_TAC [ADD_CLAUSES]
     THENL
       [ MATCH_ACCEPT_TAC
  (MATCH_MP LESS_OR (SPEC_ALL MAXWORD))
       ;
 MATCH_ACCEPT_TAC
  (MATCH_MP LESS_IMP_LESS_OR_EQ (SPEC_ALL MAXWORD))
      ]
   );;

let ALL_ONES = prove_thm
   ('ALL_ONES',
    " ! n a cin .
 ((VAL n a) + (bv cin) = 2 EXP (SUC n))
   = (ONES n a) /\ cin",
    INDUCT_TAC THEN REPEAT GEN_TAC
    THEN ASM_REWRITE_TAC [ VAL; ONES ]
    THENL
      [ REWRITE_TAC [ EXP; MULT_CLAUSES; bv11 ]
      ;
BOOL_CASES_TAC "a(SUC n):bool"
THEN REWRITE_TAC [ bvals; ADD_CLAUSES; MULT_CLAUSES ]
THENL
  [ ASM_REWRITE_TAC
      [ MULT_BY_2; SPEC "SUC n" EXP_DOUBLES;
```

```
SYM_RULE ADD_ASSOC;
(PURE_ONCE_REWRITE_RULE [ ADD_SYM ] EQ_MONO_ADD_EQ)
      ]
  ;
    ACCEPT_TAC (MATCH_MP LEQ_2_EXP (SPEC_ALL MAXWORD2))
  ]
      ]
   );;

let OVERFLOW = prove_thm
   ('OVERFLOW',
    " ! n a . (ONES n a) ==> (((VAL n a) + 1) = (2 EXP (SUC n)))",
    REPEAT STRIP_TAC
    THEN IMP_RES_TAC (snd (EQ_IMP_RULE (SPEC_ALL ALL_ONES)))
    THEN POP_ASSUM
      ( ACCEPT_TAC
         o (REWRITE_RULE [ ASSUME "ONES n a"; bvals; SYM_RULE (num_CONV "1")])
      )
   );;

let NOTOVERFLOW = prove_thm
   ('NOTOVERFLOW',
   " ! n a . ~(ONES n a /\ cin)
       ==> ((VAL n a) + (bv cin)) < (2 EXP (SUC n))",
    REPEAT GEN_TAC
    THEN PURE_REWRITE_TAC [SYM_RULE ALL_ONES]
    THEN STRIP_TAC
    THEN STRIP_ASSUME_TAC
   (PURE_ONCE_REWRITE_RULE [LESS_OR_EQ]
   (SPEC_ALL MAXWORD2))
    THEN RES_TAC
   );;


g
"!n (w:wordn) . ZEROS n w = ((VAL n w) = 0)";;

e(
REPEAT GEN_TAC
THEN EQ_TAC
THEN SPEC_TAC ("n:num","n:num")
THEN INDUCT_TAC
THEN REWRITE_TAC [ZEROS;VAL]
THEN STRIP_TAC
THEN RES_TAC
THEN ASM_REWRITE_TAC [bv;MULT_CLAUSES;ADD_CLAUSES]
);;

%

close_theory();;


%-----------------------------------------------------------------------------

   File:        busn_def.ml

   Author:      (c) D.A. Fura 1992

   Date:        14 December 1992

-----------------------------------------------------------------------------%

system 'rm busn_def.th';;

new_theory 'busn_def';;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/ml/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;
```

```
loadf 'aux_defs';;

map new_parent ['piuaux_def';'array_def';'wordn_def';'ineq'];;

new_type_abbrev('time',":num");;
new_type_abbrev('wordn',":num->bool");;

let ARBN = definition 'array_def' 'ARBN';;
let SUBARRAY_DEF = definition 'array_def' 'SUBARRAY_DEF';;

%----------------------------------------------------------------------------
   Abstract data type for a 4-valued signal.
----------------------------------------------------------------------------%

let wire =
   define_type 'wire'
               'wire = HI | LO | X | Z';;

new_type_abbrev ('busn',":num->wire");;

%----------------------------------------------------------------------------
   Type conversion functions and theorems for type WIRE.
----------------------------------------------------------------------------%

let boolVAL = new_definition
   ('boolVAL',
    "! (w :wire) .
     boolVAL w = (w = HI) => T |
                 (w = LO) => F | ARB"
   );;

let WIRE = new_definition
   ('WIRE',
    "! (b :bool) .  WIRE b = (b = T) => HI | LO"
   );;

let boolVAL_WIRE_IDENT = prove_thm
   ('boolVAL_WIRE_IDENT',
    "! (b :bool) . boolVAL (WIRE b) = b",
    REWRITE_TAC [boolVAL;WIRE]
    THEN GEN_TAC
    THEN BOOL_CASES_TAC "b:bool"
    THEN REWRITE_TAC[SYM_RULE(prove_constructors_distinct wire)]
   );;

let WIRE_boolVAL_IDENT = prove_thm
   ('WIRE_boolVAL_IDENT',
    "! (w :wire) . (w = HI) \/ (w = LO) ==> (WIRE (boolVAL w) = w)",
    REWRITE_TAC [WIRE;boolVAL]
    THEN INDUCT_THEN (prove_induction_thm wire) ASSUME_TAC
    THEN REWRITE_TAC[SYM_RULE(prove_constructors_distinct wire)]
   );;

%----------------------------------------------------------------------------
   Type conversion functions and theorems for type BUSN.
----------------------------------------------------------------------------%

let wordnVAL = new_definition
   ('wordnVAL',
    "! (f :busn) .
     wordnVAL f  =
        \(x:num). (f x = HI) => T |
                  (f x = LO) => F | ARB"
   );;

let BUSN = new_definition
   ('BUSN',
    "! (f :wordn) .
     BUSN f =
        \(x:num). (f x = T) => HI | LO"
   );;
```

```
let wordnVAL_BUSN_IDENT = prove_thm
   ('wordnVAL_BUSN_IDENT',
    "! (f :wordn) . wordnVAL (BUSN f) = f",
    REWRITE_TAC [wordnVAL;BUSN]
    THEN GEN_TAC
    THEN CONV_TAC (ONCE_DEPTH_CONV FUN_EQ_CONV)
    THEN GEN_TAC
    THEN BETA_TAC
    THEN BOOL_CASES_TAC "(f:num->bool) n"
    THEN ASM_REWRITE_TAC[SYM_RULE(prove_constructors_distinct wire)]
   );;

%----------------------------------------------------------------------------
   Special Cases.
----------------------------------------------------------------------------%

let Offn = new_definition
   ('Offn',
    "Offn = \(x:num). Z"
   );;

let wordnVAL_Offn = prove_thm
   ('wordnVAL_Offn',
    "wordnVAL Offn = ARBN",
    REWRITE_TAC [wordnVAL;Offn;ARBN;
                 SYM_RULE(prove_constructors_distinct wire)]
   );;

let OFFP = new_definition
   ('OFFP',
    "! (w :wire) .
     OFFP w = (w = Z)"
   );;

let ONP = new_definition
   ('ONP',
    "! (w :wire) .
     ONP w = ((w = HI) \/ (w = LO) \/ (w = X))"
   );;

let OFFnP = new_definition
   ('OFFnP',
    "! (f :busn) (m n :num) .
     OFFnP f (m,n) = !(x:num). (n <= x /\ x <= m) ==> (f x = Z)"
   );;

let ONnP = new_definition
   ('ONnP',
    "! (f :busn) (m n :num) .
     ONnP f (m,n) =
        !(x:num). (n <= x /\ x <= m) ==> ((f x = HI) \/ (f x = LO) \/ (f x = X))"
   );;

let OFFnP_BUSN = prove_thm
   ('OFFnP_BUSN',
    "! (f :wordn) (m n :num) . n <= m ==> (OFFnP (BUSN f) (m,n) = F)",
    REWRITE_TAC [OFFnP;BUSN]
    THEN BETA_TAC
    THEN REPEAT GEN_TAC
    THEN REWRITE_TAC [NOT_FORALL_CONV "~(!x. n <= x /\ x <= m ==>
                                    ((f x => HI | LO) = Z))"]
    THEN DISCH_TAC
    THEN EXISTS_TAC "n:num"
    THEN ASM_CASES_TAC "(f:wordn) n"
    THEN ASM_REWRITE_TAC[LESS_EQ_REFL;prove_constructors_distinct wire]
   );;

let ONnP_BUSN = prove_thm
   ('ONnP_BUSN',
    "! (f :wordn) (m n :num) . ONnP (BUSN f) (m,n) = T",
    REWRITE_TAC [ONnP;BUSN]
    THEN BETA_TAC
```

```
     THEN REPEAT STRIP_TAC
     THEN BOOL_CASES_TAC "(f:num->bool) x"
     THEN REWRITE_TAC[]
    );;

let OFFnP_Offn = prove_thm
    ('OFFnP_Offn',
     "! (m n :num) . n <= m ==> (OFFnP Offn (m,n) = T)",
     REWRITE_TAC [OFFnP;Offn]
    );;

let ONnP_Offn = prove_thm
    ('ONnP_Offn',
     "! (m n :num) . n <= m ==> (ONnP Offn (m,n) = F)",
     REWRITE_TAC [ONnP;Offn;SYM_RULE(prove_constructors_distinct wire);
                  NOT_FORALL_CONV "~(!x. ~(n <= x /\ x <= m))"]
     THEN REPEAT STRIP_TAC
     THEN EXISTS_TAC "n:num"
     THEN ASM_REWRITE_TAC [LESS_EQ_REFL]
    );;

close_theory();;


%----------------------------------------------------------------------------

   File:      templogic_def.ml

   Author:    (c) D.A. Fura 1992-93

   Date:      21 February 1993

----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/pport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

set_flag ('timing',true);;

system 'rm templogic_def.th';;

new_theory 'templogic_def';;

loadf 'aux_defs.ml';;

map new_parent ['piuaux_def';'array_def';'wordn_def';'busn_def'];;
map load_parent ['assoc';'ineq'];;

new_type_abbrev ('time',":num");;

load_library 'reduce';;

loadf 'pt_tacs.ml';;

let M_LESS_0_LESS = TAC_PROOF
    (([],
      "! m n . (m < n) ==> (0 < n)"),
      INDUCT_TAC
      THEN REPEAT STRIP_TAC
      THEN ASM_REWRITE_TAC[]
      THEN IMP_RES_TAC LT_IMP_LE
      THEN IMP_RES_TAC SUC_LE_IMP_LT
      THEN RES_TAC
    );;


%----------------------------------------------------------------------------
   Definitions involving "STABLE" boolean-valued signals.
----------------------------------------------------------------------------%
```

```
let STABLE = new_definition
   ('STABLE',
    "! (x :time->*) (t1 t2 :time) .
     STABLE x (t1,t2) =
        (t1 <= t2) /\
        (!t u :time. (t1 <= t /\ t <= t2 /\ t1 <= u /\ u <= t2) ==> (x t = x u))"
   );;

let STABLE_TRUE = new_definition
   ('STABLE_TRUE',
    "! (x :time->bool) (t1 t2 :time) .
     STABLE_TRUE x (t1,t2) =
        (t1 <= t2) /\ (!t:time. (t1 <= t /\ t <= t2) ==> (x t))"
   );;

let STABLE_FALSE = new_definition
   ('STABLE_FALSE',
    "! (x :time->bool) (t1 t2 :time) .
     STABLE_FALSE x (t1,t2) =
        (t1 <= t2) /\ (!t:time. (t1 <= t /\ t <= t2) ==> (~x t))"
   );;

let STABLE_FALSE_THEN_TRUE = new_definition
   ('STABLE_FALSE_THEN_TRUE',
    "! (x :time->bool) (t1 t2 :time) .
     STABLE_FALSE_THEN_TRUE x (t1,t2) =
        (t1 <= t2) /\ (!t:time. (t1 <= t /\ t < t2) ==> (~x t)) /\ (x t2)"
   );;

let STABLE_TRUE_THEN_FALSE = new_definition
   ('STABLE_TRUE_THEN_FALSE',
    "! (x :time->bool) (t1 t2 :time) .
     STABLE_TRUE_THEN_FALSE x (t1,t2) =
        (t1 <= t2) /\ (!t:time. (t1 <= t /\ t < t2) ==> (x t)) /\ (~x t2)"
   );;

let FALSE_THEN_STABLE_TRUE = new_definition
   ('FALSE_THEN_STABLE_TRUE',
    "! (x :time->bool) (t1 t2 :time) .
     FALSE_THEN_STABLE_TRUE x (t1,t2) =
        (t1 <= t2) /\ (~x t1) /\ (!t:time. (t1 < t /\ t <= t2) ==> (x t))"
   );;

let TRUE_THEN_STABLE_FALSE = new_definition
   ('TRUE_THEN_STABLE_FALSE',
    "! (x :time->bool) (t1 t2 :time) .
     TRUE_THEN_STABLE_FALSE x (t1,t2) =
        (t1 <= t2) /\ (x t1) /\ (!t:time. (t1 < t /\ t <= t2) ==> (~x t))"
   );;

%-----------------------------------------------------------------------------
   Definitions involving boolean-valued-signal events.
-----------------------------------------------------------------------------%

let TIME_TRUE = new_definition
   ('TIME_TRUE',
    "! (x :time->bool) (t0 t9 t :time) .
     TIME_TRUE x (t0,t9) t =
        (t0 <= t) /\ (t <= t9) /\ (x t)"
   );;

let TIME_FALSE = new_definition
   ('TIME_FALSE',
    "! (x :time->bool) (t0 t9 t :time) .
     TIME_FALSE x (t0,t9) t =
        (t0 <= t) /\ (t <= t9) /\ (~x t)"
   );;

let N_TIMES_TRUE = new_prim_rec_definition
   ('N_TIMES_TRUE',
    "(N_TIMES_TRUE 0 x t0 t9 =
```

```
        ?t:time. STABLE_FALSE_THEN_TRUE  x (t0,t) /\
                 (t < t9 ==> STABLE_FALSE  x (t+1,t9))) /\
     (N_TIMES_TRUE (SUC n) x t0 t9 =
        ?t:time. STABLE_FALSE_THEN_TRUE  x (t0,t) /\
                 N_TIMES_TRUE n x (t+1) t9)"
   );;

let N_TIMES_FALSE = new_prim_rec_definition
   ('N_TIMES_FALSE',
    "(N_TIMES_FALSE 0 x t0 t9 =
        ?t:time. STABLE_TRUE_THEN_FALSE  x (t0,t) /\
                 (t < t9 ==> STABLE_TRUE  x (t+1,t9))) /\
     (N_TIMES_FALSE (SUC n) x t0 t9 =
        ?t:time. STABLE_TRUE_THEN_FALSE  x (t0,t) /\
                 N_TIMES_FALSE n x (t+1) t9)"
   );;

let LESS_THAN_N_TIMES_TRUE = new_prim_rec_definition
   ('LESS_THAN_N_TIMES_TRUE',
    "(LESS_THAN_N_TIMES_TRUE 0 x t0 t9 = STABLE_FALSE x (t0,t9)) /\
     (LESS_THAN_N_TIMES_TRUE (SUC n) x t0 t9 =
         N_TIMES_TRUE n x t0 t9 \/
         LESS_THAN_N_TIMES_TRUE n x t0 t9)"
   );;

let LESS_THAN_N_TIMES_FALSE = new_prim_rec_definition
   ('LESS_THAN_N_TIMES_FALSE',
    "(LESS_THAN_N_TIMES_FALSE 0 x t0 t9 = STABLE_TRUE x (t0,t9)) /\
     (LESS_THAN_N_TIMES_FALSE (SUC n) x t0 t9 =
         N_TIMES_FALSE n x t0 t9 \/
         LESS_THAN_N_TIMES_FALSE n x t0 t9)"
   );;

let NTH_TIME_TRUE = new_prim_rec_definition
   ('NTH_TIME_TRUE',
    "(NTH_TIME_TRUE 0 x t0 t9 = STABLE_FALSE_THEN_TRUE x (t0,t9)) /\
     (NTH_TIME_TRUE (SUC n) x t0 t9 =
         ?(t:time) . NTH_TIME_TRUE n x t0 t /\
                     STABLE_FALSE_THEN_TRUE x (t+1,t9))"
   );;

let NTH_TIME_FALSE = new_prim_rec_definition
   ('NTH_TIME_FALSE',
    "(NTH_TIME_FALSE 0 x t0 t9 = STABLE_TRUE_THEN_FALSE x (t0,t9)) /\
     (NTH_TIME_FALSE (SUC n) x t0 t9 =
         ?(t:time) . NTH_TIME_FALSE n x t0 t /\
                     STABLE_TRUE_THEN_FALSE x (t+1,t9))"
   );;

%-----------------------------------------------------------------------------
   Definitions involving boolean-valued-signal "CHANGES."
-----------------------------------------------------------------------------%

let CHANGES_TRUE = new_definition
   ('CHANGES_TRUE',
    "! (x :time->bool) (t :time) .
     CHANGES_TRUE x t =
        ((t = 0) \/ ~x (t-1)) /\ x t"
   );;

let CHANGES_FALSE = new_definition
   ('CHANGES_FALSE',
    "! (x :time->bool) (t :time) .
     CHANGES_FALSE x t =
        ((t = 0) \/ x (t-1)) /\ ~x t"
   );;

let N_TIMES_CHANGES_TRUE = new_definition
   ('N_TIMES_CHANGES_TRUE',
    "! (n :num) (x :time->bool) (t0 t9 :time) .
     N_TIMES_CHANGES_TRUE n x t0 t9 =
        N_TIMES_TRUE n (\t. CHANGES_TRUE x t) t0 t9"
```

```
   );;

let N_TIMES_CHANGES_FALSE = new_definition
   ('N_TIMES_CHANGES_FALSE',
    "! (n :num) (x :time->bool) (t0 t9 :time) .
     N_TIMES_CHANGES_FALSE n x t0 t9 =
        N_TIMES_TRUE n (\t. CHANGES_FALSE x t) t0 t9"
   );;

let LESS_THAN_N_TIMES_CHANGES_TRUE = new_definition
   ('LESS_THAN_N_TIMES_CHANGES_TRUE',
    "! (n :num) (x :time->bool) (t0 t9 :time) .
     LESS_THAN_N_TIMES_CHANGES_TRUE n x t0 t9 =
        LESS_THAN_N_TIMES_TRUE n (\t. CHANGES_TRUE x t) t0 t9"
   );;

let LESS_THAN_N_TIMES_CHANGES_FALSE = new_definition
   ('LESS_THAN_N_TIMES_CHANGES_FALSE',
    "! (n :num) (x :time->bool) (t0 t9 :time) .
     LESS_THAN_N_TIMES_CHANGES_FALSE n x t0 t9 =
        LESS_THAN_N_TIMES_TRUE n (\t. CHANGES_FALSE x t) t0 t9"
   );;

let NTH_TIME_CHANGES_TRUE = new_prim_rec_definition
   ('NTH_TIME_CHANGES_TRUE',
    "(NTH_TIME_CHANGES_TRUE 0 x t0 t9 = STABLE_FALSE_THEN_TRUE x (t0,t9)) /\
     (NTH_TIME_CHANGES_TRUE (SUC n) x t0 t9 =
         ?t u :time.
         NTH_TIME_CHANGES_TRUE n x t0 t /\
         STABLE_TRUE_THEN_FALSE x (t,u) /\
         STABLE_FALSE_THEN_TRUE x (u,t9))"
   );;

let NTH_TIME_CHANGES_FALSE = new_prim_rec_definition
   ('NTH_TIME_CHANGES_FALSE',
    "(NTH_TIME_CHANGES_FALSE 0 x t0 t9 = STABLE_TRUE_THEN_FALSE x (t0,t9)) /\
     (NTH_TIME_CHANGES_FALSE (SUC n) x t0 t9 =
         ?t u :time.
         NTH_TIME_CHANGES_FALSE n x t0 t /\
         STABLE_FALSE_THEN_TRUE x (t,u) /\
         STABLE_TRUE_THEN_FALSE x (u,t9))"
   );;

%----------------------------------------------------------------------------
   Definitions involving "STABLE" wire-valued signals.
----------------------------------------------------------------------------%

let STABLE_HI = new_definition
   ('STABLE_HI',
    "! (x :time->wire) (t1 t2 :time) .
     STABLE_HI x (t1,t2) =
        (t1 <= t2) /\ (!t:time. (t1 <= t /\ t <= t2) ==> (x t = HI))"
   );;

let STABLE_LO = new_definition
   ('STABLE_LO',
    "! (x :time->wire) (t1 t2 :time) .
     STABLE_LO x (t1,t2) =
        (t1 <= t2) /\ (!t:time. (t1 <= t /\ t <= t2) ==> (x t = LO))"
   );;

let STABLE_LO_THEN_HI = new_definition
   ('STABLE_LO_THEN_HI',
    "! (x :time->wire) (t1 t2 :time) .
     STABLE_LO_THEN_HI x (t1,t2) =
        (t1 <= t2) /\
        (!t:time. (t1 <= t /\ t < t2) ==> (x t = LO)) /\
        (x t2 = HI)"
   );;

let STABLE_HI_THEN_LO = new_definition
   ('STABLE_HI_THEN_LO',
```

```
    "! (x :time->wire) (t1 t2 :time) .
     STABLE_HI_THEN_LO x (t1,t2) =
        (t1 <= t2) /\
        (!t:time. (t1 <= t /\ t < t2) ==> (x t = HI)) /\
        (x t2 = LO)"
   );;

let NTH_TIME_HI = new_prim_rec_definition
   ('NTH_TIME_HI',
    "(NTH_TIME_HI 0 x t0 t9 = STABLE_LO_THEN_HI x (t0,t9)) /\
     (NTH_TIME_HI (SUC n) x t0 t9 =
         ?(t:time) . NTH_TIME_HI n x t0 t /\
                     STABLE_LO_THEN_HI x (t+1,t9))"
   );;

let NTH_TIME_LO = new_prim_rec_definition
   ('NTH_TIME_LO',
    "(NTH_TIME_LO 0 x t0 t9 = STABLE_HI_THEN_LO x (t0,t9)) /\
     (NTH_TIME_LO (SUC n) x t0 t9 =
         ?(t:time) . NTH_TIME_LO n x t0 t /\
                     STABLE_HI_THEN_LO x (t+1,t9))"
   );;

%----------------------------------------------------------------------------
   Definitions involving "STABLE" (boolean- or wire-valued) signals for both
   clock phases of each cycle.
----------------------------------------------------------------------------%

let STABLE_AB = new_definition
   ('STABLE_AB',
    "! (x :time->*#*) (t1 t2 :time) .
     STABLE_AB x (t1,t2) =
       (t1 <= t2) /\
       (!t u :time.
        (t1 <= t /\ t <= t2 /\ t1 <= u /\ u <= t2) ==> (ASel(x t) = BSel(x u)))"
   );;

let STABLE_AB_TRUE = new_definition
   ('STABLE_AB_TRUE',
    "! (x :time->bool#bool) (t1 t2 :time) .
     STABLE_AB_TRUE x (t1,t2) =
        (t1 <= t2) /\
        (!t:time.
         (t1 <= t /\ t <= t2) ==> (ASel(x t) /\ BSel(x t)))"
   );;

let STABLE_AB_FALSE = new_definition
   ('STABLE_AB_FALSE',
    "! (x :time->bool#bool) (t1 t2 :time) .
     STABLE_AB_FALSE x (t1,t2) =
        (t1 <= t2) /\
        (!t:time.
         (t1 <= t /\ t <= t2) ==> (~ASel(x t) /\ ~BSel(x t)))"
   );;

let STABLE_AB_OFF = new_definition
   ('STABLE_AB_OFF',
    "! (x :time->wire#wire) (t1 t2 :time) .
     STABLE_AB_OFF x (t1,t2) =
        (t1 <= t2) /\
        (!t:time.
         (t1 <= t /\ t <= t2) ==> ((ASel(x t) = Z) /\ (BSel(x t) = Z)))"
   );;

let STABLE_AB_OFFn = new_definition
   ('STABLE_AB_OFFn',
    "! (x :time->busn#busn) (t1 t2 :time) .
     STABLE_AB_OFFn x (t1,t2) =
        (t1 <= t2) /\
        (!t:time.
         (t1 <= t /\ t <= t2) ==> ((ASel(x t) = Offn) /\ (BSel(x t) = Offn)))"
   );;
```

```
%----------------------------------------------------------------------------
   Theorems.
----------------------------------------------------------------------------%

let TRUE_EVENT_TIMES_EQUAL = prove_thm
   ('TRUE_EVENT_TIMES_EQUAL',
     "! (n :num) (t0 t1 t2 :time) (x :time->bool) .
      NTH_TIME_TRUE n x t0 t1 ==>
        NTH_TIME_TRUE n x t0 t2 ==>
           (t1 = t2)",
    INDUCT_TAC
    THEN REWRITE_TAC [NTH_TIME_TRUE;STABLE_FALSE_THEN_TRUE]
    THEN REPEAT STRIP_TAC
    THEN ASM_CASES_TAC "(t1:time) = t2"
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC NOT_EQ
    THENL [
       SPEC_ASSUM_TAC ("!t. t0 <= t /\ t < t2 ==> ~x t","t1:time")
       THEN RES_TAC
    ;
       SPEC_ASSUM_TAC ("!t. t0 <= t /\ t < t1 ==> ~x t","t2:time")
       THEN RES_TAC
    ;
       RES_TAC
       THEN UNDISCH_TAC "(t' + 1) <= t1 ==> ~x t1"
       THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "(t':time) = t") []
       THEN ASM_REWRITE_TAC[]
    ;
       RES_TAC
       THEN UNDISCH_TAC "(t + 1) <= t2 ==> ~x t2"
       THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "(t:time) = t'") []
       THEN ASM_REWRITE_TAC[]
    ]
   );;

let FALSE_EVENT_TIMES_EQUAL = prove_thm
   ('FALSE_EVENT_TIMES_EQUAL',
     "! (n :num) (t0 t1 t2 :time) (x :time->bool) .
      NTH_TIME_FALSE n x t0 t1 ==>
        NTH_TIME_FALSE n x t0 t2 ==>
           (t1 = t2)",
    INDUCT_TAC
    THEN REWRITE_TAC [NTH_TIME_FALSE;STABLE_TRUE_THEN_FALSE]
    THEN REPEAT STRIP_TAC
    THEN ASM_CASES_TAC "(t1:time) = t2"
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC NOT_EQ
    THENL [
       SPEC_ASSUM_TAC ("!t. t0 <= t /\ t < t2 ==> x t","t1:time")
       THEN RES_TAC
       THEN RES_TAC
    ;
       SPEC_ASSUM_TAC ("!t. t0 <= t /\ t < t1 ==> x t","t2:time")
       THEN RES_TAC
       THEN RES_TAC
    ;
       RES_TAC
       THEN UNDISCH_TAC "(t' + 1) <= t1 ==> x t1"
       THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "(t':time) = t") []
       THEN ASM_REWRITE_TAC[]
    ;
       RES_TAC
       THEN UNDISCH_TAC "(t + 1) <= t2 ==> x t2"
       THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "(t:time) = t'") []
       THEN ASM_REWRITE_TAC[]
    ]
   );;

 let STABLE_TRUE_THEN_FALSE_UNIQUE = prove_thm
    ('STABLE_TRUE_THEN_FALSE_UNIQUE',
     "! (t0 t1 t2 :time) (x :time->bool) .
```

```
     STABLE_TRUE_THEN_FALSE x (t0,t1) ==>
       STABLE_TRUE_THEN_FALSE x (t0,t2) ==>
         (t1 = t2)",
    REWRITE_TAC [STABLE_TRUE_THEN_FALSE]
    THEN REPEAT STRIP_TAC
    THEN ASM_CASES_TAC "(t1:time) = t2"
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC NOT_EQ
    THENL [
       SPEC_ASSUM_TAC ("!t. t0 <= t /\ t < t2 ==> x t","t1:time")
       THEN RES_TAC
       THEN RES_TAC
    ;
       SPEC_ASSUM_TAC ("!t. t0 <= t /\ t < t1 ==> x t","t2:time")
       THEN RES_TAC
       THEN RES_TAC
    ]
   );;

let TRUE_EVENT_TIMES_MONO = prove_thm
   ('TRUE_EVENT_TIMES_MONO',
    "! (n :num) (t0 t1 t2 :time) (x :time->bool) .
     NTH_TIME_TRUE n x t0 t1 ==>
       NTH_TIME_TRUE (SUC n) x t0 t2 ==>
         (t1 < t2)",
    REWRITE_TAC [NTH_TIME_TRUE;STABLE_FALSE_THEN_TRUE]
    THEN REPEAT STRIP_TAC
    THEN IMP_RES_TAC TRUE_EVENT_TIMES_EQUAL
    THEN IMP_RES_TAC (REWRITE_RULE [ADD1] OR_LESS)
    THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "t1:time = t") []
    THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "t < t2") []
   );;

let SINGLE_TRUE_INTERVAL_EVENT_TIMES_EQUAL = prove_thm
   ('SINGLE_TRUE_INTERVAL_EVENT_TIMES_EQUAL',
    "! (t0 t9 t u :time) (x :time->bool) .
     N_TIMES_TRUE 0 x t0 t9 ==>
       TIME_TRUE x (t0,t9) t ==>
         TIME_TRUE x (t0,t9) u ==>
           (t = u)",
    REWRITE_TAC [TIME_TRUE;N_TIMES_TRUE;STABLE_FALSE;STABLE_FALSE_THEN_TRUE]
    THEN REPEAT STRIP_TAC
    THEN ASM_CASES_TAC "t' < t9"
    THENL [
       % subgoal 1: [ "t' < t9" ] %
       IMP_RES_TAC (REWRITE_RULE [ADD1] LESS_OR)
       THEN RES_TAC
       THEN NRULE_ASSUM_TAC
               ("(t' + 1) <= u ==> ~x u",((REWRITE_RULE []) o CONTRAPOS))
       THEN NRULE_ASSUM_TAC
               ("(t' + 1) <= t ==> ~x t",((REWRITE_RULE []) o CONTRAPOS))
       THEN NRULE_ASSUM_TAC
               ("u < t' ==> ~x u", ((REWRITE_RULE []) o CONTRAPOS))
       THEN NRULE_ASSUM_TAC
               ("t < t' ==> ~x t", ((REWRITE_RULE []) o CONTRAPOS))
       THEN RES_TAC
       THEN IMP_RES_TAC NOT_LESS_EQ_LESS
       THEN IMP_RES_TAC NOT_LESS
       THEN IMP_RES_TAC SUB_LESS_OR
       THEN ASSUME_TAC (REWRITE_RULE [] (REDUCE_CONV "1<=1"))
       THEN IMP_RES_TAC (SPECL ["t':time";"1";"1"] ASSOC_ADD_SUB1)
       THEN ASM_REWRITE_ASSUM_TAC ("t <= ((t' + 1) - 1)",
                                   [SUB_EQUAL_0;ADD_CLAUSES])
       THEN ASM_REWRITE_ASSUM_TAC ("u <= ((t' + 1) - 1)",
                                   [SUB_EQUAL_0;ADD_CLAUSES])
       THEN IMP_RES_TAC LESS_EQUAL_ANTISYM
       THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "u:time = t'") []
       THEN FILTER_ASM_REWRITE_TAC (\tm. tm = "t:time = t'") []
    ;
       % subgoal 2:  [ "~t' < t9" ] %
       IMP_RES_TAC NOT_LESS
       THEN RES_TAC
```

```
        THEN IMP_RES_TAC LESS_EQ_TRANS
        THEN REWRITE_ASSUM_TAC ("u <= t'",[LESS_OR_EQ])
        THEN REWRITE_ASSUM_TAC ("t <= t'",[LESS_OR_EQ])
        THEN POP_ASSUM_LIST (MAP_EVERY (\thm. STRIP_ASSUME_TAC thm))
        THEN RES_TAC
        THEN ASM_REWRITE_TAC[]
    ]
   );;

let LATER_TRUE_EVENT_FOLLOWS_INTERVAL = prove_thm
   ('LATER_TRUE_EVENT_FOLLOWS_INTERVAL',
    "! (t0 t9 t' t'suc :time) (x :time->bool) .
     N_TIMES_TRUE 0 x t0 t9 ==>
       t0 <= t' ==>
         x t' ==>
           t' < t'suc ==>
             x t'suc ==>
               t9 < t'suc",
    REWRITE_TAC [STABLE_FALSE;STABLE_FALSE_THEN_TRUE]
    THEN REPEAT STRIP_TAC
    THEN ASM_CASES_TAC "t' <= t9"
    THENL [
       % subgoal 1: [ "t' <= t9" ] %
       ASM_CASES_TAC "t9 < t'suc"
       THEN ASM_REWRITE_TAC[]
       THEN REWRITE_ASSUM_TAC ("~t9 < t'suc",[NOT_LESS])
       THEN IMP_RES_TAC LESS_EQ_LESS_TRANS
       THEN IMP_RES_TAC LT_IMP_LE
       THEN IMP_RES_TAC
               (SPECL ["t0:time";"t9:time";"t':time";"t'suc:time"]
                      (REWRITE_RULE [TIME_TRUE]
                                    SINGLE_TRUE_INTERVAL_EVENT_TIMES_EQUAL))
       THEN IMP_RES_TAC LESS_NOT_EQ
    ;
       % subgoal 2: [ "~t' <= t9" ] %
       REWRITE_ASSUM_TAC ("~t' <= t9",[NOT_LESS_EQ_LESS])
       THEN IMP_RES_TAC LESS_TRANS
       THEN ASM_REWRITE_TAC[]
    ]
   );;

let SUB_STABLE_FALSE_THEN_TRUE = prove_thm
   ('SUB_STABLE_FALSE_THEN_TRUE',
    "! (t1 t2 t9 :time) (x :time->bool) .
     STABLE_FALSE_THEN_TRUE x (t1,t9) ==>
       (t1 <= t2) ==>
         (t2 <= t9) ==>
           STABLE_FALSE_THEN_TRUE x (t2,t9)",
    REWRITE_TAC [STABLE_FALSE_THEN_TRUE]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC LESS_EQ_TRANS
    THEN SPEC_ASSUM_TAC ("!t. t1 <= t /\ t < t9 ==> ~x t","t:time")
    THEN RES_TAC
   );;

let SUB_INTERVALS_STABLE_TRUE = prove_thm
   ('SUB_INTERVALS_STABLE_TRUE',
    "! (t0 t9 t1 :time) (x :time->bool) .
     STABLE_TRUE x (t0,t9) ==>
       (t0 <= t1) ==>
         (t1 <= t9) ==>
           (STABLE_TRUE x (t0,t1) /\ STABLE_TRUE x (t1,t9))",
    REWRITE_TAC [STABLE_TRUE]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC LESS_EQ_TRANS
    THEN RES_TAC
   );;

let SUB_INTERVALS_STABLE_FALSE = prove_thm
   ('SUB_INTERVALS_STABLE_FALSE',
```

```
    "! (t0 t9 t1 :time) (x :time->bool) .
     STABLE_FALSE x (t0,t9) ==>
        (t0 <= t1) ==>
           (t1 <= t9) ==>
              (STABLE_FALSE x (t0,t1) /\ STABLE_FALSE x (t1,t9))",
    REWRITE_TAC [STABLE_FALSE]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC LESS_EQ_TRANS
    THEN RES_TAC
   );;

let STABLE_FALSE_THEN = prove_thm
   ('STABLE_FALSE_THEN',
    "! (x :time->bool) (t0 t1 :time) .
     STABLE_FALSE_THEN_TRUE x (t0,t1) ==>
       t0 < t1 ==>
         STABLE_FALSE x (t0,t1-1)",
    REWRITE_TAC [STABLE_FALSE_THEN_TRUE;STABLE_FALSE]
    THEN REPEAT STRIP_TAC
    THEN IMP_RES_TAC (REWRITE_RULE [PRE_SUB1] LT_IMP_LE_PRE)
    THEN ASM_REWRITE_TAC[]
    THEN IMP_RES_TAC M_LESS_0_LESS
    THEN IMP_RES_TAC (REWRITE_RULE [ADD1] LT_IMP_SUC_LE)
    THEN POP_ASSUM_LIST (MAP_EVERY (\thm. ASSUME_TAC (REDUCE_RULE thm)))
    THEN IMP_RES_TAC (REWRITE_RULE [PRE_SUB1] LE_PRE_IMP_LT)
    THEN RES_TAC
   );;

let SUP_INTERVAL_STABLE_TRUE = mk_thm
   ([],
    "! (t0 t9 t1 t2 :time) (x :time->bool) .
     STABLE_TRUE x (t0,t1) ==>
        STABLE_TRUE x (t2,t9) ==>
           (t2 <= t1+1) ==>
              (t0 <= t9) ==>
                 STABLE_TRUE x (t0,t9)"
   );;

let SUP_INTERVAL_STABLE_FALSE = mk_thm
   ([],
    "! (t0 t9 t1 t2 :time) (x :time->bool) .
     STABLE_FALSE x (t0,t1) ==>
        STABLE_FALSE x (t2,t9) ==>
           (t2 <= t1+1) ==>
              (t0 <= t9) ==>
                 STABLE_FALSE x (t0,t9)"
   );;

let SUP_INTERVAL_STABLE_TRUE_THEN_FALSE = prove_thm
   ('SUP_INTERVAL_STABLE_TRUE_THEN_FALSE',
    "! (t0 t9 t1 t2 :time) (x :time->bool) .
     STABLE_TRUE x (t0,t1) ==>
        STABLE_TRUE_THEN_FALSE x (t2,t9) ==>
           (t2 <= t1+1) ==>
              (t0 <= t9) ==>
                 STABLE_TRUE_THEN_FALSE x (t0,t9)",
    REWRITE_TAC [STABLE_TRUE;STABLE_TRUE_THEN_FALSE]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
    THEN ASSUME_TAC
           (REWRITE_RULE
              [STABLE_TRUE]
              (SPECL ["t0:time";"t1:time";"t2:time"] SUB_INTERVALS_STABLE_TRUE))      THEN RES_TAC
    THEN ASM_CASES_TAC "t2 <= t"
    THEN RES_TAC
    THEN REWRITE_ASSUM_TAC ("~t2 <= t",[NOT_LESS_EQ_LESS])
    THEN IMP_RES_TAC (SPECL ["t2:time";"t1+1";"1"] LESS_EQ_MONO_SUB)
    THEN ASSUME_TAC (SPEC "1" LESS_EQ_REFL)
    THEN IMP_RES_TAC (SPECL ["t1:time";"1";"1"] ASSOC_ADD_SUB1)
    THEN ASM_REWRITE_ASSUM_TAC
           ("(t2 - 1) <= ((t1 + 1) - 1)",[SUB_EQUAL_0;ADD_CLAUSES])
```

```
    THEN IMP_RES_TAC SUB_LESS_OR
    THEN IMP_RES_TAC LESS_EQ_TRANS
    THEN RES_TAC
   );;

let SUP_INTERVAL_STABLE_FALSE_THEN_TRUE = prove_thm
   ('SUP_INTERVAL_STABLE_FALSE_THEN_TRUE',
    "! (t0 t9 t1 t2 :time) (x :time->bool) .
     STABLE_FALSE x (t0,t1) ==>
        STABLE_FALSE_THEN_TRUE x (t2,t9) ==>
           (t2 <= t1+1) ==>
              (t0 <= t9) ==>
                 STABLE_FALSE_THEN_TRUE x (t0,t9)",
    REWRITE_TAC [STABLE_FALSE;STABLE_FALSE_THEN_TRUE]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
    THEN ASSUME_TAC
           (REWRITE_RULE
              [STABLE_FALSE]
              (SPECL ["t0:time";"t1:time";"t2:time"] SUB_INTERVALS_STABLE_FALSE))    THEN RES_TAC
    THEN ASM_CASES_TAC "t2 <= t"
    THEN RES_TAC
    THEN REWRITE_ASSUM_TAC ("-t2 <= t",[NOT_LESS_EQ_LESS])
    THEN IMP_RES_TAC (SPECL ["t2:time";"t1+1";"1"] LESS_EQ_MONO_SUB)
    THEN ASSUME_TAC (SPEC "1" LESS_EQ_REFL)
    THEN IMP_RES_TAC (SPECL ["t1:time";"1";"1"] ASSOC_ADD_SUB1)
    THEN ASM_REWRITE_ASSUM_TAC
           ("(t2 - 1) <= ((t1 + 1) - 1)",[SUB_EQUAL_0;ADD_CLAUSES])
    THEN IMP_RES_TAC SUB_LESS_OR
    THEN IMP_RES_TAC LESS_EQ_TRANS
    THEN RES_TAC
   );;

close_theory();;


%-----------------------------------------------------------------------------

File:       gates_def1.ml

Author:     (c) D.A. Fura 1992-93

Date:       4 March 1993

This file contains the ml source for the combinational logic gates used in the
gate-level description of the FTEP PIU, an ASIC developed by the Embedded
Processing Laboratory, Boeing High Technology Center.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing', true);;

system 'rm gates_def1.th';;

new_theory 'gates_def1';;

map new_parent ['piuaux_def';'wordn_def';'busn_def';'ineq'];;

let NOT_GATE = new_definition
('NOT_GATE',
 "! a z :time->bool#bool.
  NOT_GATE a z =
  !t:time . z t = ((-ASel(a t)), (-BSel(a t)))"
);;

let AND2_GATE = new_definition
('AND2_GATE',
 "! a b z :time->bool#bool.
```

```
  AND2_GATE a b z =
  !t:time . z t = ((ASel(a t) /\ ASel(b t)),
 (BSel(a t) /\ BSel(b t)))"
);;

let AND3_GATE = new_definition
('AND3_GATE',
 "! a b c z :time->bool#bool.
  AND3_GATE a b c z =
  !t:time . z t = ((ASel(a t) /\ ASel(b t) /\ ASel(c t)),
 (BSel(a t) /\ BSel(b t) /\ BSel(c t)))"
);;

let OR2_GATE = new_definition
('OR2_GATE',
 "! a b z :time->bool#bool.
  OR2_GATE a b z =
  !t:time . z t = ((ASel(a t) \/ ASel(b t)),
 (BSel(a t) \/ BSel(b t)))"
);;

let OR3_GATE = new_definition
('OR3_GATE',
 "! a b c z :time->bool#bool.
  OR3_GATE a b c z =
  !t:time . z t = ((ASel(a t) \/ ASel(b t) \/ ASel(c t)),
 (BSel(a t) \/ BSel(b t) \/ BSel(c t)))"
);;

let NAND2_GATE = new_definition
('NAND2_GATE',
 "! a b z :time->bool#bool.
  NAND2_GATE a b z =
  !t:time . z t = ((~(ASel(a t) /\ ASel(b t))),
 (~(BSel(a t) /\ BSel(b t))))"
);;

let NAND3_GATE = new_definition
('NAND3_GATE',
 "! a b c z :time->bool#bool.
  NAND3_GATE a b c z =
  !t:time . z t = ((~(ASel(a t) /\ ASel(b t) /\ ASel(c t))),
 (~(BSel(a t) /\ BSel(b t) /\ BSel(c t))))"
);;

let BUF_GATE = new_definition
('BUF_GATE',
 "! a z :time->*#*.
  BUF_GATE a z =
  !t:time . z t = (ASel(a t), BSel(a t))"
);;

let TRIBUF_GATE = new_definition
('TRIBUF_GATE',
 "! (z :time->wire#wire) (a e :time->bool#bool) .
  TRIBUF_GATE a e z =
  ! t:time . z t = ((ASel(e t) => WIRE (ASel(a t)) | Z),
                    (BSel(e t) => WIRE (BSel(a t)) | Z))"
);;

let TRIBUFn_GATE = new_definition
('TRIBUFn_GATE',
 "! (a :time->wordn#wordn) (z :time->busn#busn) (e :time->bool#bool) .
  TRIBUFn_GATE a e z =
  ! t:time . z t = ((ASel(e t) => BUSN (ASel(a t)) | Offn),
                    (BSel(e t) => BUSN (BSel(a t)) | Offn))"
);;

let TRINBUF_GATE = new_definition
('TRINBUF_GATE',
 "! (z :time->wire#wire) (a e :time->bool#bool) .
  TRINBUF_GATE a e z =
```

```
  ! t:time . z t = (((~ASel(e t)) => WIRE (ASel(a t)) | Z),
                          ((~BSel(e t)) => WIRE (BSel(a t)) | Z))"
);;

let TRINBUFn_GATE = new_definition
('TRINBUFn_GATE',
 "! (a :time->wordn#wordn) (z :time->busn#busn) (e :time->bool#bool) .
  TRINBUFn_GATE a e z =
  ! t:time . z t = (((~ASel(e t)) => BUSN (ASel(a t)) | Offn),
                          ((~BSel(e t)) => BUSN (BSel(a t)) | Offn))"
);;

close_theory();;


%----------------------------------------------------------------------------

File:latches_def.ml

Author:(c) D.A. Fura 1992

Date:31 August 1992

This file contains the ml source for the latches used in the gate-level
specification of the FTEP PIU, an ASIC developed by the Embedded Processing
Laboratory, Boeing High Technology Center.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/']);;

set_flag ('timing', true);;

system 'rm latches_def.th';;

new_theory 'latches_def';;

map new_parent ['piuaux_def'];;

%----------------------------------------------------------------------------
One-bit A-clocked D-latch, no set, no reset, no enable.
-----------------------------------------------------------------------------%

let DLatA_GATE = new_definition
('DLatA_GATE',
 "! (d q :time->bool#bool) (s :time->bool).
  DLatA_GATE d s q =
! t:time .
(s (t+1) = ASel(d t)) /\
(q t = (s (t+1), s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit B-clocked D-latch, no set, no reset, no enable.
-----------------------------------------------------------------------------%

let DLatB_GATE = new_definition
('DLatB_GATE',
 "! (d q :time->bool#bool) (s :time->bool).
  DLatB_GATE d s q =
! t:time .
(s (t+1) = BSel(d t)) /\
(q t = (s t, s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit A-clocked D-latch, with set, no reset, no enable.
-----------------------------------------------------------------------------%

let DSLatA_GATE = new_definition
('DSLatA_GATE',
 "! (d set q :time->bool#bool) (s :time->bool) .
```

```
  DSLatA_GATE d set s q =
! t:time .
(s (t+1) = (ASel(set t)) => T | ASel(d t)) /\
(q t = (s (t+1), s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit B-clocked D-latch, with set, no reset, no enable.
-----------------------------------------------------------------------------%

let DSLatB_GATE = new_definition
('DSLatB_GATE',
 "! (d set q :time->bool#bool) (s :time->bool) .
  DSLatB_GATE d set s q =
! t:time .
(s (t+1) = (BSel(set t)) => T | BSel(d t)) /\
(q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit A-clocked D-latch, no set, with reset, no enable.
-----------------------------------------------------------------------------%

let DRLatA_GATE = new_definition
('DRLatA_GATE',
 "! (d rst q :time->bool#bool) (s :time->bool) .
  DRLatA_GATE d rst s q =
! t:time .
(s (t+1) = (ASel(rst t)) => F | ASel(d t)) /\
(q t = (s (t+1), s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit B-clocked D-latch, no set, with reset, no enable.
-----------------------------------------------------------------------------%

let DRLatB_GATE = new_definition
('DRLatB_GATE',
 "! (d rst q :time->bool#bool) (s :time->bool) .
  DRLatB_GATE d rst s q =
! t:time .
(s (t+1) = (BSel(rst t)) => F | BSel(d t)) /\
(q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit A-clocked D-latch, with set, with reset, no enable.
-----------------------------------------------------------------------------%

let DSRLatA_GATE = new_definition
('DSRLatA_GATE',
 "! (d set rst q :time->bool#bool) (s :time->bool) .
  DSRLatA_GATE d set rst s q =
! t:time .
(s (t+1) = ((ASel(set t) /\ ~ASel(rst t)) => T |
((~ASel(set t)) /\ ASel(rst t)) => F |
((~ASel(set t)) /\ ~ASel(rst t)) => ASel(d t) |
ARB)) /\
(q t = (s (t+1), s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit B-clocked D-latch, with set, with reset, no enable.
-----------------------------------------------------------------------------%

let DSRLatB_GATE = new_definition
('DSRLatB_GATE',
 "! (d set rst q :time->bool#bool) (s :time->bool) .
  DSRLatB_GATE d set rst s q =
! t:time .
(s (t+1) = ((BSel(set t) /\ ~BSel(rst t)) => T |
((~BSel(set t)) /\ BSel(rst t)) => F |
```

```
((~BSel(set t)) /\ ~BSel(rst t)) => BSel(d t) |
ARB)) /\
(q t = (s t, s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit A-clocked D-latch, no set, no reset, with enable.
----------------------------------------------------------------------------%

let DELatA_GATE = new_definition
('DELatA_GATE',
 "! (d en q :time->bool#bool) (s :time->bool) .
  DELatA_GATE d en s q =
! t:time .
(s (t+1) = (ASel(en t)) => ASel(d t) | s t) /\
(q t = (s (t+1), s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit B-clocked D-latch, no set, no reset, with enable.
----------------------------------------------------------------------------%

let DELatB_GATE = new_definition
('DELatB_GATE',
 "! (d en q :time->bool#bool) (s :time->bool) .
  DELatB_GATE d en s q =
! t:time .
(s (t+1) = (BSel(en t)) => BSel(d t) | s t) /\
(q t = (s t, s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit A-clocked D-latch, no set, with reset, with enable.
----------------------------------------------------------------------------%

let DRELatA_GATE = new_definition
('DRELatA_GATE',
 "! (d rst en q :time->bool#bool) (s :time->bool) .
  DRELatA_GATE d rst en s q =
! t:time .
(s (t+1) = (ASel(en t)) => ((ASel(rst t)) => F | ASel(d t)) | s t) /\
(q t = (s (t+1), s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit B-clocked D-latch, no set, with reset, with enable.
----------------------------------------------------------------------------%

let DRELatB_GATE = new_definition
('DRELatB_GATE',
 "! (d rst en q :time->bool#bool) (s :time->bool) .
  DRELatB_GATE d rst en s q =
! t:time .
(s (t+1) = (BSel(en t)) => ((BSel(rst t)) => F | BSel(d t)) | s t) /\
(q t = (s t, s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit A-clocked D-latch, with set, no reset, with enable.
----------------------------------------------------------------------------%

let DSELatA_GATE = new_definition
('DSELatA_GATE',
 "! (d set en q :time->bool#bool) (s :time->bool) .
  DSELatA_GATE d set en s q =
! t:time .
(s (t+1) = (ASel(en t)) => ((ASel(set t)) => T | ASel(d t)) | s t) /\
(q t = (s (t+1), s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit B-clocked D-latch, with set, no reset, with enable.
```

```
------------------------------------------------------------------------------%

let DSELatB_GATE = new_definition
('DSELatB_GATE',
 "! (d set en q :time->bool#bool) (s :time->bool) .
  DSELatB_GATE d set en s q =
! t:time .
(s (t+1) = (BSel(en t)) => ((BSel(set t)) => T | BSel(d t)) | s t) /\
(q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit A-clocked D-latch, with set, with reset, with enable.
------------------------------------------------------------------------------%

let DSRELatA_GATE = new_definition
('DSRELatA_GATE',
 "! (d set rst en q :time->bool#bool) (s :time->bool) .
  DSRELatA_GATE d set rst en s q =
! t:time .
(s (t+1) = (ASel(en t))
                    => ((ASel(set t) /\ ~ASel(rst t)) => T |
      ((~ASel(set t)) /\ ASel(rst t)) => F |
      ((~ASel(set t)) /\ ~ASel(rst t)) => ASel(d t) |
      ARB)
  | s t) /\
(q t = (s (t+1), s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit B-clocked D-latch, with set, with reset, with enable.
------------------------------------------------------------------------------%

let DSRELatB_GATE = new_definition
('DSRELatB_GATE',
 "! (d set rst en q :time->bool#bool) (s :time->bool) .
  DSRELatB_GATE d set rst en s q =
! t:time .
(s (t+1) = (BSel(en t))
                    => ((BSel(set t) /\ ~BSel(rst t)) => T |
      ((~BSel(set t)) /\ BSel(rst t)) => F |
      ((~BSel(set t)) /\ ~BSel(rst t)) => BSel(d t) |
      ARB)
  | s t) /\
(q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
N-bit A-clocked D-latch, no set, no reset, no enable.
------------------------------------------------------------------------------%

let DLatNA_GATE = new_definition
('DLatNA_GATE',
 "! (d q :time->wordn#wordn) (s :time->wordn) .
  DLatNA_GATE d s q =
! t:time .
(s (t+1) = ASel(d t)) /\
(q t = (s (t+1), s (t+1)))"
);;

%-----------------------------------------------------------------------------
N-bit B-clocked D-latch, no set, no reset, no enable.
------------------------------------------------------------------------------%

let DLatNB_GATE = new_definition
('DLatNB_GATE',
 "! (d q :time->wordn#wordn) (s :time->wordn) .
  DLatNB_GATE d s q =
! t:time .
(s (t+1) = BSel(d t)) /\
(q t = (s t, s (t+1)))"
);;
```

```
close_theory();;

%-----------------------------------------------------------------------------

File:        ffs_def.ml

Author:      (c) D.A. Fura 1992

Date:        21 September 1992

This file contains the ml source for the flip-flops used in the gate-level
specification of the FTEP PIU, an ASIC developed by the Embedded Processing
Laboratory, Boeing High Technology Center.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/']);;

set_flag ('timing', true);;

system 'rm ffs_def.th';;

new_theory 'ffs_def';;

map new_parent ['piuaux_def'];;

%-----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, no set, no reset, no enable.
-----------------------------------------------------------------------------%

let DFFA_GATE = new_definition
('DFFA_GATE',
 "! (d q :time->bool#bool) (s :time->bool) .
  DFFA_GATEd s q =
     ! t:time .
             (s (t+1) = BSel(d t)) /\
             (q t = (s t, s t))"
);;

%-----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, no set, no reset, no enable.
-----------------------------------------------------------------------------%

let DFFB_GATE = new_definition
('DFFB_GATE',
 "! (d q :time->bool#bool) (s :time->bool) .
  DFFB_GATEd s q =
     ! t:time .
             (s (t+1) = ASel(d t)) /\
             (q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, no set, with reset, no enable.
-----------------------------------------------------------------------------%

let DRFFA_GATE = new_definition
('DRFFA_GATE',
 "! (d rst q :time->bool#bool) (s :time->bool) .
  DRFFA_GATE d rst s q =
     ! t:time .
             (s (t+1) = BSel(rst t) => F | BSel(d t)) /\
             (q t = (s t, s t))"
);;

%-----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, no set, with reset, no enable.
-----------------------------------------------------------------------------%

let DRFFB_GATE = new_definition
```

```
('DRFFB_GATE',
 "! (d rst q :time->bool#bool) (s :time->bool) .
  DRFFB_GATE d rst s q =
     ! t:time .
             (s (t+1) = ASel(rst t) => F | ASel(d t)) /\
             (q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, with set, no reset, no enable.
-----------------------------------------------------------------------------%

let DSFFA_GATE = new_definition
('DSFFA_GATE',
 "! (d set q :time->bool#bool) (s :time->bool) .
  DSFFA_GATE d set s q =
     ! t:time .
             (s (t+1) = BSel(set t) => T | BSel(d t)) /\
             (q t = (s t, s t))"
);;

%-----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, with set, no reset, no enable.
-----------------------------------------------------------------------------%

let DSFFB_GATE = new_definition
('DSFFB_GATE',
 "! (d set q :time->bool#bool) (s :time->bool) .
  DSFFB_GATE d set s q =
     ! t:time .
             (s (t+1) = ASel(set t) => T | ASel(d t)) /\
             (q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, with set, with reset, no enable.
-----------------------------------------------------------------------------%

let DRSFFA_GATE = new_definition
('DRSFFA_GATE',
 "! (d rst set q :time->bool#bool) (s :time->bool) .
  DRSFFA_GATE d rst set s q =
  ! t:time .
             (s (t+1) = (BSel(set t) /\ ~BSel(rst t)) => T |
                        ((~BSel(set t)) /\ BSel(rst t)) => F |
                        ((~BSel(set t)) /\ ~BSel(rst t)) => BSel(d t) |
                        ARB) /\
             (q t = (s t, s t))"
);;

%-----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, with set, with reset, no enable.
-----------------------------------------------------------------------------%

let DRSFFB_GATE = new_definition
('DRSFFB_GATE',
 "! (d rst set q :time->bool#bool) (s :time->bool) .
  DRSFFB_GATE d rst set s q =
  ! t:time .
             (s (t+1) = (ASel(set t) /\ ~ASel(rst t)) => T |
                        ((~ASel(set t)) /\ ASel(rst t)) => F |
                        ((~ASel(set t)) /\ ~ASel(rst t)) => BSel(d t) |
                        ARB) /\
             (q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, no set, no reset, with enable.
-----------------------------------------------------------------------------%

let DEFFA_GATE = new_definition
   ('DEFFA_GATE',
```

```
    "! (d en q :time->bool#bool) (s :time->bool) .
     DEFFA_GATE d en s q =
        ! t:time .
             (s (t+1) = BSel(en t) => BSel(d t) | s t) /\
             (q t = (s t, s t))"
);;

%-----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, no set, no reset, with enable.
-----------------------------------------------------------------------------%

let DEFFB_GATE = new_definition
   ('DEFFB_GATE',
    "! (d en q :time->bool#bool) (s :time->bool) .
     DEFFB_GATE d en s q =
        ! t:time .
             (s (t+1) = ASel(en t) => ASel(d t) | s t) /\
             (q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
N-bit positive-triggered flip-flop, no set, no reset, with enable.
-----------------------------------------------------------------------------%

let DEFFnA_GATE = new_definition
   ('DEFFnA_GATE',
    "! (d q :time->wordn#wordn) (en :time->bool#bool) (s :time->wordn) .
     DEFFnA_GATE d en s q =
        ! t:time .
             (s (t+1) = BSel(en t) => BSel(d t) | s t) /\
             (q t = (s t, s t))"
);;

%-----------------------------------------------------------------------------
N-bit negative-triggered flip-flop, no set, no reset, with enable.
-----------------------------------------------------------------------------%

let DEFFnB_GATE = new_definition
   ('DEFFnB_GATE',
    "! (d q :time->wordn#wordn) (en :time->bool#bool) (s :time->wordn) .
     DEFFnB_GATE d en s q =
        ! t:time .
             (s (t+1) = ASel(en t) => ASel(d t) | s t) /\
             (q t = (s t, s (t+1)))"
);;

%-----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, no set, with reset, with enable.
-----------------------------------------------------------------------------%

let DREFFA_GATE = new_definition
   ('DREFFA_GATE',
    "! (d en rst q :time->bool#bool) (s :time->bool) .
     DREFFA_GATE d en rst s q =
        ! t:time .
             (s (t+1) = BSel(en t) => (BSel(rst t) => F | BSel(d t)) | s t) /\
             (q t = (s t, s t))"
);;

%-----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, no set, with reset, with enable.
-----------------------------------------------------------------------------%

let DREFFB_GATE = new_definition
   ('DREFFB_GATE',
    "! (d en rst q :time->bool#bool) (s :time->bool) .
     DREFFB_GATE d en rst s q =
        ! t:time .
             (s (t+1) = ASel(en t) => (ASel(rst t) => F | ASel(d t)) | s t) /\
             (q t = (s t, s (t+1)))"
);;
```

```
%----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, with set, no reset, with enable.
----------------------------------------------------------------------------%

let DSEFFA_GATE = new_definition
   ('DSEFFA_GATE',
    "! (d en rst q :time->bool#bool) (s :time->bool) .
     DSEFFA_GATE d en rst s q =
        ! t:time .
             (s (t+1) = BSel(en t) => (BSel(rst t) => T | BSel(d t)) | s t) /\
             (q t = (s t, s t))"
);;

%----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, with set, no reset, with enable.
----------------------------------------------------------------------------%

let DSEFFB_GATE = new_definition
   ('DSEFFB_GATE',
    "! (d en rst q :time->bool#bool) (s :time->bool) .
     DSEFFB_GATE d en rst s q =
        ! t:time .
             (s (t+1) = ASel(en t) => (ASel(rst t) => T | ASel(d t)) | s t) /\
             (q t = (s t, s (t+1)))"
);;

%----------------------------------------------------------------------------
One-bit positive-triggered flip-flop, with set, with reset, with enable.
----------------------------------------------------------------------------%

let DRSEFFA_GATE = new_definition
   ('DRSEFFA_GATE',
    "! (d en rst set q :time->bool#bool) (s :time->bool) .
     DRSEFFA_GATE d en rst set s q =
        ! t:time .
             (s (t+1) = BSel(en t)
                 => ((BSel(set t) /\ ~BSel(rst t)) => T |
                      (~BSel(set t) /\ BSel(rst t)) => F |
                      (~BSel(set t) /\ ~BSel(rst t)) => BSel(d t) | ARB)
                 | s t) /\
             (q t = (s t, s t))"
);;

%----------------------------------------------------------------------------
One-bit negative-triggered flip-flop, with set, with reset, with enable.
----------------------------------------------------------------------------%

let DRSEFFB_GATE = new_definition
   ('DRSEFFB_GATE',
    "! (d en rst set q :time->bool#bool) (s :time->bool) .
     DRSEFFB_GATE d en rst set s q =
        ! t:time .
             (s (t+1) = ASel(en t)
                 => ((ASel(set t) /\ ~ASel(rst t)) => T |
                      (~ASel(set t) /\ ASel(rst t)) => F |
                      (~ASel(set t) /\ ~ASel(rst t)) => ASel(d t) | ARB)
                 | s t) /\
             (q t = (s t, s (t+1)))"
);;

close_theory();;


%----------------------------------------------------------------------------

File:        counters_def.ml

Author:      (c) D.A. Fura 1992-93

Date:        4 March 1993

This file contains the ml source for the counters used in the gate-level
```

```
specification of the FTEP PIU, an ASIC developed by the Embedded Processing
Laboratory, Boeing High Technology Center.

------------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing', true);;

system 'rm counters_def.th';;

new_theory 'counters_def';;

map new_parent ['piuaux_def';'wordn_def';'array_def';'ineq'];;

%-----------------------------------------------------------------------------
Positive-triggered up-counter, no reset.
------------------------------------------------------------------------------%

let UpCntA_GATE = new_definition
('UpCntA_GATE',
 "! (sz :num) (d q :time->wordn#wordn) (ld up z :time->bool#bool)
       (s :time->wordn) .
     UpCntA_GATE sz d ld up s q z =
        ! t:time .
             (s (t+1) = (BSel(ld t)) => BSel(d t) |
                        (BSel(up t)) => INCN sz (s t) | s t) /\
             (q t = (((ASel(up t)) => INCN sz (s t) | s t) ,
                     ((BSel(up t)) => INCN sz (s t) | s t))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

%-----------------------------------------------------------------------------
Negative-triggered up-counter, no reset.
------------------------------------------------------------------------------%

let UpCntB_GATE = new_definition
('UpCntB_GATE',
 "! (sz :num) (d q :time->wordn#wordn) (ld up z :time->bool#bool)
       (s :time->wordn) .
     UpCntB_GATE sz d ld up s q z =
        ! t:time .
             (s (t+1) = (ASel(ld t)) => ASel(d t) |
                        (ASel(up t)) => INCN sz (s t) | s t) /\
             (q t = (((ASel(up t)) => INCN sz (s t) | s t) ,
                     ((BSel(up t)) => INCN sz (s (t+1)) | s (t+1)))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

%-----------------------------------------------------------------------------
Positive-triggered down-counter, no reset.
------------------------------------------------------------------------------%

let DownCntA_GATE = new_definition
('DownCntA_GATE',
 "! (sz :num) (d q :time->wordn#wordn) (ld dn z :time->bool#bool)
       (s :time->wordn) .
     DownCntA_GATE sz d ld dn s q z =
        ! t:time .
             (s (t+1) = (BSel(ld t)) => BSel(d t) |
                        (BSel(dn t)) => DECN sz (s t) | s t) /\
             (q t = (((ASel(dn t)) => DECN sz (s t) | s t) ,
                     ((BSel(dn t)) => DECN sz (s t) | s t))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

%-----------------------------------------------------------------------------
Negative-triggered down-counter, no reset.
------------------------------------------------------------------------------%
```

```
let DownCntB_GATE = new_definition
('DownCntB_GATE',
 "! (sz :num) (d q :time->wordn#wordn) (ld dn z :time->bool#bool)
       (s :time->wordn) .
     DownCntB_GATE sz d ld dn s q z =
        ! t:time .
             (s (t+1) = (ASel(ld t)) => ASel(d t) |
                        (ASel(dn t)) => DECN sz (s t) | s t) /\
             (q t = (((ASel(dn t)) => DECN sz (s t) | s t) ,
                     ((BSel(dn t)) => DECN sz (s (t+1)) | s (t+1)))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

%----------------------------------------------------------------------------
Positive-triggered up-counter, with reset.
----------------------------------------------------------------------------%

let UpRCntA_GATE = new_definition
   ('UpRCntA_GATE',
    "! (sz :num) (d q :time->wordn#wordn) (ld up rst z :time->bool#bool)
       (s :time->wordn) .
     UpRCntA_GATE sz d ld up rst s q z =
        ! t:time .
             (s (t+1) = (BSel(rst t)) => WORDN sz 0 |
                        (BSel(ld t)) => BSel(d t) |
                        (BSel(up t)) => INCN sz (s t) | s t) /\
             (q t = (((ASel(up t)) => INCN sz (s t) | s t) ,
                     ((BSel(up t)) => INCN sz (s t) | s t))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

%----------------------------------------------------------------------------
Negative-triggered up-counter, with reset.
----------------------------------------------------------------------------%

let UpRCntB_GATE = new_definition
   ('UpRCntB_GATE',
    "! (sz :num) (d q :time->wordn#wordn) (ld up rst z :time->bool#bool)
       (s :time->wordn) .
     UpRCntB_GATE sz d ld up rst s q z =
        ! t:time .
             (s (t+1) = (ASel(rst t)) => WORDN sz 0 |
                        (ASel(ld t)) => ASel(d t) |
                        (ASel(up t)) => INCN sz (s t) | s t) /\
             (q t = (((ASel(up t)) => INCN sz (s t) | s t) ,
                     ((BSel(up t)) => INCN sz (s (t+1)) | s (t+1)))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

%----------------------------------------------------------------------------
Positive-triggered down-counter, with reset.
----------------------------------------------------------------------------%

let DownRCntA_GATE = new_definition
   ('DownRCntA_GATE',
    "! (sz :num) (d q :time->wordn#wordn) (ld dn rst z :time->bool#bool)
       (s :time->wordn) .
     DownRCntA_GATE sz d ld dn rst s q z =
        ! t:time .
             (s (t+1) = (BSel(rst t)) => WORDN sz 0 |
                        (BSel(ld t)) => BSel(d t) |
                        (BSel(dn t)) => DECN sz (s t) | s t) /\
             (q t = (((ASel(dn t)) => DECN sz (s t) | s t) ,
                     ((BSel(dn t)) => DECN sz (s t) | s t))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

%----------------------------------------------------------------------------
Negative-triggered down-counter, with reset.
----------------------------------------------------------------------------%

let DownRCntB_GATE = new_definition
```

```
   ('DownRCntB_GATE',
    "! (sz :num) (d q :time->wordn#wordn) (ld dn rst z :time->bool#bool)
       (s :time->wordn) .
     DownRCntB_GATE sz d ld dn rst s q z =
        ! t:time .
             (s (t+1) = (ASel(rst t)) => WORDN sz 0 |
                        (ASel(ld t)) => ASel(d t) |
                        (ASel(dn t)) => DECN sz (s t) | s t) /\
             (q t = (((ASel(dn t)) => DECN sz (s t) | s t) ,
                     ((BSel(dn t)) => DECN sz (s (t+1)) | s (t+1)))) /\
             (z t = ((ASel(q t) = WORDN sz 0), (BSel(q t) = WORDN sz 0)))"
);;

close_theory();;


%----------------------------------------------------------------------------

   File:        datapaths_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        4 March 1993

   This file contains the ml source for the datapath blocks of the R-Port of the
   FTEP PIU, an ASIC developed by the Embedded Processing Laboratory, Boeing High
   Technology Center.

----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/'
                                 ]);;

set_flag ('timing', true);;

system 'rm datapaths_def.th';;

new_theory 'datapaths_def';;

loadf 'abs_theory';;

map new_parent ['piuaux_def';'array_def';'wordn_def';'ineq'];;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%----------------------------------------------------------------------------
   Counter block used to build timers.
----------------------------------------------------------------------------%

let DP_CTR_GATE = new_definition
   ('DP_CTR_GATE',
    "! (busB_in busA_out1 busA_out2 :time->wordn#wordn)
       (cir_wr c_ld cir_rd ce cin csror_ld cor_rd c_out :time->bool#bool)
       (r_ctr_in r_ctr r_ctr_new r_ctr_out :time->wordn)
       (r_ctr_mux_sel r_ctr_irden r_ctr_ce r_ctr_cin r_ctr_cry
        r_ctr_orden :time->bool) .
     DP_CTR_GATE busB_in cir_wr c_ld cir_rd ce cin csror_ld cor_rd
                 r_ctr_in r_ctr_mux_sel r_ctr_irden r_ctr r_ctr_ce r_ctr_cin
                 r_ctr_cry
                 r_ctr_new r_ctr_out r_ctr_orden busA_out1 busA_out2 c_out =
        !(t:time).
        (r_ctr_in (t+1) = (BSel(cir_wr t)) => BSel(busB_in t) | r_ctr_in t) /\
        (r_ctr_mux_sel (t+1) = BSel(c_ld t)) /\
        (r_ctr_irden (t+1) = BSel(cir_rd t)) /\
        (r_ctr (t+1) = (r_ctr_mux_sel t) => r_ctr_in t | r_ctr_new t) /\
        (r_ctr_ce (t+1) = ASel(ce t)) /\
        (r_ctr_cin (t+1) = ASel(cin t)) /\
        (r_ctr_cry (t+1) = (r_ctr_ce t) /\ (r_ctr_cin t) /\ ONES 31 (r_ctr t)) /\
        (r_ctr_new (t+1) =
           ((r_ctr_ce t) /\ (r_ctr_cin t)) => INCN 31 (r_ctr t) | r_ctr t) /\
```

```
        (r_ctr_out (t+1) = (BSel(csror_ld t)) => r_ctr_new t | r_ctr_out t) /\
        (r_ctr_orden (t+1) = BSel(cor_rd t)) /\
        (busA_out1 t =
           (((r_ctr_irden t) => r_ctr_in t | ARBN), ARBN)) /\
        (busA_out2 t =
           (((r_ctr_orden t) => r_ctr_out t | ARBN), ARBN)) /\
        (c_out t = ((r_ctr_cry t), (r_ctr_cry (t+1))))"
   );;

%-----------------------------------------------------------------------------
   Interrupt Control Register (ICR) block.
-----------------------------------------------------------------------------%

let DP_ICR_GATE = new_definition
   ('DP_ICR_GATE',
    "! (rep :^REP_ty)
       (busA_in busB_in busA_out icr_out :time->wordn#wordn)
       (icr_wr_feedback icr_wr icr_select icr_ld icr_rd :time->bool#bool)
       (r_icr_old r_icr_mask r_icr :time->wordn)
       (r_icr_rden :time->bool) .
     DP_ICR_GATE rep busA_in busB_in icr_wr_feedback icr_wr icr_select icr_ld icr_rd
                 r_icr_old r_icr_mask r_icr r_icr_rden busA_out icr_out =
        !(t:time).
        (r_icr_old (t+1) =
           (BSel(icr_wr_feedback t)) => BSel(busA_in t) | r_icr_old t) /\
        (r_icr_mask (t+1) =
           (BSel(icr_wr t)) => BSel(busB_in t) | r_icr_mask t) /\
        (r_icr (t+1) =
           (BSel(icr_ld t))
              => (ASel(icr_select t)) => Andn rep (r_icr_old t, r_icr_mask t)
                                       | Orn rep (r_icr_old t, r_icr_mask t)
                 | r_icr t) /\
        (r_icr_rden (t+1) = BSel(icr_rd t)) /\
        (busA_out t = (((r_icr_rden t) => r_icr t | ARBN), ARBN)) /\
        (icr_out t = (r_icr t, r_icr (t+1)))"
   );;

%-----------------------------------------------------------------------------
   Control register used to build General Control Register (GCR) and Communication
   Control Register (CCR).
-----------------------------------------------------------------------------%

let DP_CR_GATE = new_definition
   ('DP_CR_GATE',
    "! (busB_in busA_out cr_out :time->wordn#wordn)
       (cr_wr cr_rd :time->bool#bool)
       (r_cr :time->wordn)
       (r_cr_rden :time->bool) .
     DP_CR_GATE busB_in cr_wr cr_rd
                r_cr r_cr_rden
                busA_out cr_out =
        !(t:time).
        (r_cr (t+1) = (BSel(cr_wr t)) => BSel(busB_in t) | r_cr t) /\
        (r_cr_rden (t+1) = BSel(cr_rd t)) /\
        (busA_out t = (((r_cr_rden t) => r_cr t | ARBN), ARBN)) /\
        (cr_out t = (r_cr t, r_cr (t+1)))"
   );;

%-----------------------------------------------------------------------------
   Status Register Block.
-----------------------------------------------------------------------------%

let DP_SR_GATE = new_definition
   ('DP_SR_GATE',
    "! (inp busA_out :time->wordn#wordn)
       (sror_ld sr_rd :time->bool#bool)
       (r_sr :time->wordn)
       (r_sr_rden :time->bool) .
     DP_SR_GATE inp sror_ld sr_rd
                r_sr r_sr_rden
                busA_out =
        !(t:time).
```

```
        (r_sr (t+1) = (BSel(sror_ld t)) => BSel(inp t) | r_sr t) /\
        (r_sr_rden (t+1) = BSel(sr_rd t)) /\
        (busA_out t = (((r_sr_rden t) => r_sr t | ARBN), ARBN))"
   );;

close_theory();;


%----------------------------------------------------------------------------

   File:        buses_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        4 March 1993

----------------------------------------------------------------------------%

system 'rm buses_def.th';;

new_theory 'buses_def';;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/ml/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

map new_parent ['piuaux_def';'array_def';'wordn_def';'busn_def';'ineq'];;

new_type_abbrev('time',":num");;
new_type_abbrev('wordn',":num->bool");;

let Bus2n_CF = new_definition
   ('Bus2n_CF',
    "! (m n :num) (inD1 inD2 :busn#busn) .
     Bus2n_CF (m,n) inD1 inD2 =
        let offa1 = OFFnP (ASel inD1) (m,n) in
        let offa2 = OFFnP (ASel inD2) (m,n) in
        let offb1 = OFFnP (BSel inD1) (m,n) in
        let offb2 = OFFnP (BSel inD2) (m,n) in
        (((~offa1) => offa2 | T) /\
         ((~offb1) => offb2 | T))"
   );;

let Bus12n_CF = new_definition
   ('Bus12n_CF',
    "! (m n :num) (inD1 inD2 inD3 inD4 inD5 inD6 inD7 inD8 inD9 inD10 inD11
        inD12 :busn#busn) .
     Bus12n_CF (m,n) inD1 inD2 inD3 inD4 inD5 inD6 inD7 inD8 inD9 inD10 inD11
               inD12 =
        let offa1 = OFFnP (ASel inD1) (m,n) in
        let offa2 = OFFnP (ASel inD2) (m,n) in
        let offa3 = OFFnP (ASel inD3) (m,n) in
        let offa4 = OFFnP (ASel inD4) (m,n) in
        let offa5 = OFFnP (ASel inD5) (m,n) in
        let offa6 = OFFnP (ASel inD6) (m,n) in
        let offa7 = OFFnP (ASel inD7) (m,n) in
        let offa8 = OFFnP (ASel inD8) (m,n) in
        let offa9 = OFFnP (ASel inD9) (m,n) in
        let offa10 = OFFnP (ASel inD10) (m,n) in
        let offa11 = OFFnP (ASel inD11) (m,n) in
        let offa12 = OFFnP (ASel inD12) (m,n) in
        let offb1 = OFFnP (BSel inD1) (m,n) in
        let offb2 = OFFnP (BSel inD2) (m,n) in
        let offb3 = OFFnP (BSel inD3) (m,n) in
        let offb4 = OFFnP (BSel inD4) (m,n) in
        let offb5 = OFFnP (BSel inD5) (m,n) in
        let offb6 = OFFnP (BSel inD6) (m,n) in
        let offb7 = OFFnP (BSel inD7) (m,n) in
        let offb8 = OFFnP (BSel inD8) (m,n) in
        let offb9 = OFFnP (BSel inD9) (m,n) in
        let offb10 = OFFnP (BSel inD10) (m,n) in
```

```
        let offb11 = OFFnP (BSel inD11) (m,n) in
        let offb12 = OFFnP (BSel inD12) (m,n) in
        (((~offa1) => (offa2 /\ offa3 /\ offa4 /\ offa5 /\ offa6 /\ offa7 /\
                       offa8 /\ offa9 /\ offa10 /\ offa11 /\ offa12) |
          (~offa2) => (offa3 /\ offa4 /\ offa5 /\ offa6 /\ offa7 /\ offa8 /\
                       offa9 /\ offa10 /\ offa11 /\ offa12) |
          (~offa3) => (offa4 /\ offa5 /\ offa6 /\ offa7 /\ offa8 /\ offa9 /\
                       offa10 /\ offa11 /\ offa12) |
          (~offa4) => (offa5 /\ offa6 /\ offa7 /\ offa8 /\ offa9 /\ offa10 /\
                       offa11 /\ offa12) |
          (~offa5) => (offa6 /\ offa7 /\ offa8 /\ offa9 /\ offa10 /\ offa11 /\
                       offa12) |
          (~offa6) => (offa7 /\ offa8 /\ offa9 /\ offa10 /\ offa11 /\ offa12) |
          (~offa7) => (offa8 /\ offa9 /\ offa10 /\ offa11 /\ offa12) |
          (~offa8) => (offa9 /\ offa10 /\ offa11 /\ offa12) |
          (~offa9) => (offa10 /\ offa11 /\ offa12) |
          (~offa10) => (offa11 /\ offa12) |
          (~offa11) => (offa12) | T) /\
         ((~offb1) => (offb2 /\ offb3 /\ offb4 /\ offb5 /\ offb6 /\ offb7 /\
                       offb8 /\ offb9 /\ offb10 /\ offb11 /\ offb12) |
          (~offb2) => (offb3 /\ offb4 /\ offb5 /\ offb6 /\ offb7 /\ offb8 /\
                       offb9 /\ offb10 /\ offb11 /\ offb12) |
          (~offb3) => (offb4 /\ offb5 /\ offb6 /\ offb7 /\ offb8 /\ offb9 /\
                       offb10 /\ offb11 /\ offb12) |
          (~offb4) => (offb5 /\ offb6 /\ offb7 /\ offb8 /\ offb9 /\ offb10 /\
                       offb11 /\ offb12) |
          (~offb5) => (offb6 /\ offb7 /\ offb8 /\ offb9 /\ offb10 /\ offb11 /\
                       offb12) |
          (~offb6) => (offb7 /\ offb8 /\ offb9 /\ offb10 /\ offb11 /\ offb12) |
          (~offb7) => (offb8 /\ offb9 /\ offb10 /\ offb11 /\ offb12) |
          (~offb8) => (offb9 /\ offb10 /\ offb11 /\ offb12) |
          (~offb9) => (offb10 /\ offb11 /\ offb12) |
          (~offb10) => (offb11 /\ offb12) |
          (~offb11) => (offb12) | T))"
   );;

let MERGE2n_GATE = new_definition
   ('MERGE2n_GATE',
    "! (m n :num) (inD1 inD2 out :time->busn#busn) .
     MERGE2n_GATE (m,n) inD1 inD2 out =
        !t:time.
        out t =
           (((Bus2n_CF (m,n) (inD1 t) (inD2 t))
                => (ONnP (ASel(inD1 t)) (m,n)) => (ASel(inD1 t)) |
                   (ONnP (ASel(inD2 t)) (m,n)) => (ASel(inD2 t)) | Offn
                | ARBN),
            ((Bus2n_CF (m,n) (inD1 t) (inD2 t))
                => (ONnP (BSel(inD1 t)) (m,n)) => (BSel(inD1 t)) |
                   (ONnP (BSel(inD2 t)) (m,n)) => (BSel(inD2 t)) | Offn
                | ARBN))"
   );;

let JOIN2n_GATE = new_definition
   ('JOIN2n_GATE',
    "! (m n :num) (inD1 inD2 :time->busn#busn) (out :time->wordn#wordn) .
     JOIN2n_GATE (m,n) inD1 inD2 out =
        !t:time.
        out t =
           (((Bus2n_CF (m,n) (inD1 t) (inD2 t))
               => (ONnP (ASel(inD1 t)) (m,n)) => wordnVAL (ASel(inD1 t)) |
                  (ONnP (ASel(inD2 t)) (m,n)) => wordnVAL (ASel(inD2 t))
                                               | wordnVAL (Offn)
               | ARBN),
            ((Bus2n_CF (m,n) (inD1 t) (inD2 t))
               => (ONnP (BSel(inD1 t)) (m,n)) => wordnVAL (BSel(inD1 t)) |
                  (ONnP (BSel(inD2 t)) (m,n)) => wordnVAL (BSel(inD2 t))
                                               | wordnVAL (Offn)
               | ARBN))"
   );;

let JOIN12n_GATE = new_definition
   ('JOIN12n_GATE',
```

```
"! (m n :num) (inD1 inD2 inD3 inD4 inD5 inD6 inD7 inD8 inD9 inD10 inD11
    inD12 :time->busn#busn) (out :time->wordn#wordn) .
 JOIN12n_GATE (m,n) inD1 inD2 inD3 inD4 inD5 inD6 inD7 inD8 inD9 inD10 inD11
              inD12 out =
    !t:time.
    out t =
       (((Bus12n_CF (m,n) (inD1 t) (inD2 t) (inD3 t) (inD4 t) (inD5 t)
                    (inD6 t) (inD7 t) (inD8 t) (inD9 t) (inD10 t) (inD11 t)
                    (inD12 t))
          => (ONnP (ASel(inD1 t)) (m,n)) => wordnVAL (ASel(inD1 t)) |
             (ONnP (ASel(inD2 t)) (m,n)) => wordnVAL (ASel(inD2 t)) |
             (ONnP (ASel(inD3 t)) (m,n)) => wordnVAL (ASel(inD3 t)) |
             (ONnP (ASel(inD4 t)) (m,n)) => wordnVAL (ASel(inD4 t)) |
             (ONnP (ASel(inD5 t)) (m,n)) => wordnVAL (ASel(inD5 t)) |
             (ONnP (ASel(inD6 t)) (m,n)) => wordnVAL (ASel(inD6 t)) |
             (ONnP (ASel(inD7 t)) (m,n)) => wordnVAL (ASel(inD7 t)) |
             (ONnP (ASel(inD8 t)) (m,n)) => wordnVAL (ASel(inD8 t)) |
             (ONnP (ASel(inD9 t)) (m,n)) => wordnVAL (ASel(inD9 t)) |
             (ONnP (ASel(inD10 t)) (m,n)) => wordnVAL (ASel(inD10 t)) |
             (ONnP (ASel(inD11 t)) (m,n)) => wordnVAL (ASel(inD11 t)) |
             (ONnP (ASel(inD12 t)) (m,n)) => wordnVAL (ASel(inD12 t))
                                             | wordnVAL (Offn)
          | ARBN),
        ((Bus12n_CF (m,n) (inD1 t) (inD2 t) (inD3 t) (inD4 t) (inD5 t)
                    (inD6 t) (inD7 t) (inD8 t) (inD9 t) (inD10 t) (inD11 t)
                    (inD12 t))
          => (ONnP (BSel(inD1 t)) (m,n)) => wordnVAL (BSel(inD1 t)) |
             (ONnP (BSel(inD2 t)) (m,n)) => wordnVAL (BSel(inD2 t)) |
             (ONnP (BSel(inD3 t)) (m,n)) => wordnVAL (BSel(inD3 t)) |
             (ONnP (BSel(inD4 t)) (m,n)) => wordnVAL (BSel(inD4 t)) |
             (ONnP (BSel(inD5 t)) (m,n)) => wordnVAL (BSel(inD5 t)) |
             (ONnP (BSel(inD6 t)) (m,n)) => wordnVAL (BSel(inD6 t)) |
             (ONnP (BSel(inD7 t)) (m,n)) => wordnVAL (BSel(inD7 t)) |
             (ONnP (BSel(inD8 t)) (m,n)) => wordnVAL (BSel(inD8 t)) |
             (ONnP (BSel(inD9 t)) (m,n)) => wordnVAL (BSel(inD9 t)) |
             (ONnP (BSel(inD10 t)) (m,n)) => wordnVAL (BSel(inD10 t)) |
             (ONnP (BSel(inD11 t)) (m,n)) => wordnVAL (BSel(inD11 t)) |
             (ONnP (BSel(inD12 t)) (m,n)) => wordnVAL (BSel(inD12 t))
                                             | wordnVAL (Offn)
          | ARBN))"
  );;

close_theory();;
```

3 PIU Design Specification

This section contains the HOL listings for the PIU design specification. Subsection 3.1 contains definitions used throughout the PIU specification. Subsections 3.2–3.6 contain the specifications for the P-Port, M-Port, R-Port, C-Port, and SU-Cont, respectively. Each of these subsections contains three theories, defining the data stuctures, the gate-level structure, and the clock-level behavior for each of the five PIU ports.

3.1 PIU-Applicable Definitions

This section contains the code for the theory *piuaux_def*, containing several definitions used throughout the PIU specification.

```
%-----------------------------------------------------------------------------

   File:        piuaux_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        1 March 1993

   This file contains auxiliary definitions needed for the specification of the
   FTEP PIU, an ASIC developed by the Embedded Processing Laboratory, Boeing
   High Technology Center.

-----------------------------------------------------------------------------%

set_flag ('timing', true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/'
                                 ]);;

system 'rm piuaux_def.th';;

new_theory 'piuaux_def';;

loadf 'abs_theory';;

new_type_abbrev('time', ":num");;
new_type_abbrev('wordn', ":(num->bool)");;

%-----------------------------------------------------------------------------
   Abstract data type for the SU_Cont FSM states.
-----------------------------------------------------------------------------%

let sfsm_ty_Axiom =
   define_type 'sfsm_ty_Axiom'
               'sfsm_ty = SSTART | SRA | SPF | SC0I | SC0F | ST | SC1I |
                          SC1F | SS | SSTOP | SCS | SN | SO';;

let ASel = new_definition
   ('ASel',
    "!x:*#* . ASel x = FST x"
   );;

let BSel = new_definition
   ('BSel',
    "!x:*#* . BSel x = SND x"
   );;

let sig = new_definition
   ('sig',
    "sig (sel :*->**) (signal :time->*) = (\t. (sel (signal t)))"
   );;
```

```
let asig = new_definition
   ('asig',
    "asig (sel :*->**#**) (signal :time->*) = (\t. ASel (sel (signal t)))"
   );;

let bsig = new_definition
   ('bsig',
    "bsig (sel :*->**#**) (signal :time->*) = (\t. BSel (sel (signal t)))"
   );;

let VDD = new_definition
   ('VDD',
    "! t:time . VDD t = T,T"
   );;

let GND = new_definition
   ('GND',
    "! t:time . GND t = F,F"
   );;

let REP_lemma = new_abstract_representation 'REP'
   [('Andn', ":(wordn#wordn->wordn)");
    ('Orn', ":(wordn#wordn->wordn)");
    ('Ham_Dec', ":(wordn->wordn)");
    ('Ham_Det1', ":(wordn->wordn)");
    ('Ham_Det2', ":(wordn#bool->bool)");
    ('Ham_Enc', ":(wordn->wordn)");
    ('Par_Dec', ":(wordn->wordn)");
    ('Par_Det', ":(wordn->bool)");
    ('Par_Enc', ":(wordn->wordn)")
   ];;

let rep_ty = abs_type_info REP_lemma;;

close_theory();;
```

3.2 P-Port Definitions

This section contains the theories *paux_def*, *pblock_def*, and *pclock_def*, defining the P-Port design.

```
%-----------------------------------------------------------------------------

   File:        paux_def.ml

   Author:      (c) D.A. Fura 1992

   Date:        10 December 1992

-----------------------------------------------------------------------------%

set_flag ('timing', true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftop/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/work/'
                                 ]);;

system 'rm paux_def.th';;

new_theory 'paux_def';;

map new_parent ['busn_def'];;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('timeC', ":num");;
new_type_abbrev ('wordn', ":(num->bool)");;
new_type_abbrev ('busn', ":(num->wire)");;
```

```
%-----------------------------------------------------------------------------
   Abstract data type for the P-Port FSM states.
-----------------------------------------------------------------------------%
let pfsm_ty_Axiom =
   define_type 'pfsm_ty_Axiom'
               'pfsm_ty = PH | PA | PD';;

%-----------------------------------------------------------------------------
   Abstract data type for the P-Port instruction.
-----------------------------------------------------------------------------%
let PCI =
   define_type 'PCI'
               'PCI = PC_X';;

%-----------------------------------------------------------------------------
   Abstract data type for the state.
-----------------------------------------------------------------------------%
let pc_state =
   define_type 'pc_state'
               'pc_state = PCState wordn bool wordn bool pfsm_ty bool bool bool
                           bool bool bool bool bool wordn bool bool bool
                           bool bool bool';;

let P_addrS = new_recursive_definition
    false
    pc_state
    'P_addrS'
    "P_addrS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                  P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                  P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                  P_male_ P_rale_)
     = P_addr";;

let P_dest1S = new_recursive_definition
    false
    pc_state
    'P_dest1S'
    "P_dest1S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                  P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                  P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                  P_male_ P_rale_)
     = P_dest1";;

let P_be_S = new_recursive_definition
    false
    pc_state
    'P_be_S'
    "P_be_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                  P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                  P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                  P_male_ P_rale_)
     = P_be_";;

let P_wrS = new_recursive_definition
    false
    pc_state
    'P_wrS'
    "P_wrS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                  P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                  P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                  P_male_ P_rale_)
     = P_wr";;

let P_fsm_stateS = new_recursive_definition
    false
    pc_state
    'P_fsm_stateS'
    "P_fsm_stateS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                  P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                  P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                  P_male_ P_rale_)
     = P_fsm_state";;
```

```
let P_fsm_rstS = new_recursive_definition
    false
    pc_state
    'P_fsm_rstS'
    "P_fsm_rstS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                 P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                 P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                 P_male_ P_rale_)
     = P_fsm_rst";;

let P_fsm_mrqtS = new_recursive_definition
    false
    pc_state
    'P_fsm_mrqtS'
    "P_fsm_mrqtS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                  P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                  P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                  P_male_ P_rale_)
     = P_fsm_mrqt";;

let P_fsm_sackS = new_recursive_definition
    false
    pc_state
    'P_fsm_sackS'
    "P_fsm_sackS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                  P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                  P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                  P_male_ P_rale_)
     = P_fsm_sack";;

let P_fsm_cgnt_S = new_recursive_definition
    false
    pc_state
    'P_fsm_cgnt_S'
    "P_fsm_cgnt_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_fsm_cgnt_";;

let P_fsm_crqt_S = new_recursive_definition
    false
    pc_state
    'P_fsm_crqt_S'
    "P_fsm_crqt_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_fsm_crqt_";;

let P_fsm_hold_S = new_recursive_definition
    false
    pc_state
    'P_fsm_hold_S'
    "P_fsm_hold_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_fsm_hold_";;

let P_fsm_lock_S = new_recursive_definition
    false
    pc_state
    'P_fsm_lock_S'
    "P_fsm_lock_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_fsm_lock_";;

let P_rqtS = new_recursive_definition
```

```
    false
    pc_state
    'P_rqtS'
    "P_rqtS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_rqt";;

let P_sizeS = new_recursive_definition
    false
    pc_state
    'P_sizeS'
    "P_sizeS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_size";;

let P_loadS = new_recursive_definition
    false
    pc_state
    'P_loadS'
    "P_loadS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_load";;

let P_downS = new_recursive_definition
    false
    pc_state
    'P_downS'
    "P_downS (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_down";;

let P_lock_S = new_recursive_definition
    false
    pc_state
    'P_lock_S'
    "P_lock_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_lock_";;

let P_lock_inh_S = new_recursive_definition
    false
    pc_state
    'P_lock_inh_S'
    "P_lock_inh_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_lock_inh_";;

let P_male_S = new_recursive_definition
    false
    pc_state
    'P_male_S'
    "P_male_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                   P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                   P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                   P_male_ P_rale_)
     = P_male_";;

let P_rale_S = new_recursive_definition
    false
    pc_state
```

```
    'P_rale_S'
    "P_rale_S (PCState P_addr P_dest1 P_be_ P_wr P_fsm_state P_fsm_rst
                    P_fsm_mrqt P_fsm_sack P_fsm_cgnt_ P_fsm_crqt_ P_fsm_hold_
                    P_fsm_lock_ P_rqt P_size P_load P_down P_lock_ P_lock_inh_
                    P_male_ P_rale_)
     = P_rale_";;

let State_CASES =
   prove_cases_thm (prove_induction_thm pc_state);;

let State_Selectors_Work = prove_thm
   ('State_Selectors_Work',
    "!s:pc_state .
     s = (PCState (P_addrS s) (P_dest1S s) (P_be_S s) (P_wrS s) (P_fsm_stateS s)
         (P_fsm_rstS s) (P_fsm_mrqtS s) (P_fsm_sackS s) (P_fsm_cgnt_S s)
         (P_fsm_crqt_S s) (P_fsm_hold_S s) (P_fsm_lock_S s) (P_rqtS s)
         (P_sizeS s) (P_loadS s) (P_downS s) (P_lock_S s) (P_lock_inh_S s)
         (P_male_S s) (P_rale_S s))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "s:pc_state" State_CASES)
    THEN REWRITE_TAC [P_addrS; P_dest1S; P_be_S; P_wrS; P_fsm_stateS; P_fsm_rstS;
                      P_fsm_mrqtS; P_fsm_sackS; P_fsm_cgnt_S; P_fsm_crqt_S;
                      P_fsm_hold_S; P_fsm_lock_S; P_rqtS; P_sizeS; P_loadS;
                      P_downS; P_lock_S; P_lock_inh_S; P_male_S; P_rale_S]
   );;


%----------------------------------------------------------------------------
   Abstract data type for the environment.
----------------------------------------------------------------------------%

let pc_env =
   define_type 'pc_env'
               'pc_env = PCEnv bool#bool wordn#wordn bool#bool bool#bool
                               wordn#wordn bool#bool bool#bool wordn#wordn
                               bool#bool bool#bool bool#bool';;

let RstE = new_recursive_definition
    false
    pc_env
    'RstE'
    "RstE (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
                 I_cgnt_ I_hold_ I_srdy_)
     = Rst";;

let L_ad_inE = new_recursive_definition
    false
    pc_env
    'L_ad_inE'
    "L_ad_inE (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
                 I_cgnt_ I_hold_ I_srdy_)
     = L_ad_in";;

let L_ads_E = new_recursive_definition
    false
    pc_env
    'L_ads_E'
    "L_ads_E (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
                 I_cgnt_ I_hold_ I_srdy_)
     = L_ads_";;

let L_den_E = new_recursive_definition
    false
    pc_env
    'L_den_E'
    "L_den_E (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
                 I_cgnt_ I_hold_ I_srdy_)
     = L_den_";;

let L_be_E = new_recursive_definition
    false
    pc_env
```

```
    'L_be_E'
    "L_be_E (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
              I_cgnt_ I_hold_ I_srdy_)
     = L_be_";;

let L_wrE = new_recursive_definition
    false
    pc_env
    'L_wrE'
    "L_wrE (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
              I_cgnt_ I_hold_ I_srdy_)
     = L_wr";;

let L_lock_E = new_recursive_definition
    false
    pc_env
    'L_lock_E'
    "L_lock_E (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
              I_cgnt_ I_hold_ I_srdy_)
     = L_lock_";;

let I_ad_inE = new_recursive_definition
    false
    pc_env
    'I_ad_inE'
    "I_ad_inE (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
              I_cgnt_ I_hold_ I_srdy_)
     = I_ad_in";;

let I_cgnt_E = new_recursive_definition
    false
    pc_env
    'I_cgnt_E'
    "I_cgnt_E (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
              I_cgnt_ I_hold_ I_srdy_)
     = I_cgnt_";;

let I_hold_E = new_recursive_definition
    false
    pc_env
    'I_hold_E'
    "I_hold_E (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
              I_cgnt_ I_hold_ I_srdy_)
     = I_hold_";;

let I_srdy_E = new_recursive_definition
    false
    pc_env
    'I_srdy_E'
    "I_srdy_E (PCEnv Rst L_ad_in L_ads_ L_den_ L_be_ L_wr L_lock_ I_ad_in
              I_cgnt_ I_hold_ I_srdy_)
     = I_srdy_";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm pc_env);;

let Env_Selectors_Work = prove_thm
   ('Env_Selectors_Work',
    "!e:pc_env .
      e = (PCEnv (RstE e) (L_ad_inE e) (L_ads_E e) (L_den_E e) (L_be_E e)
              (L_wrE e) (L_lock_E e) (I_ad_inE e) (I_cgnt_E e) (I_hold_E e)
              (I_srdy_E e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:pc_env" Env_CASES)
    THEN REWRITE_TAC [RstE; L_ad_inE; L_ads_E; L_den_E; L_be_E; L_wrE; L_lock_E;
                      I_ad_inE; I_cgnt_E; I_hold_E; I_srdy_E]
   );;

%----------------------------------------------------------------------------
   Abstract data type for the output.
----------------------------------------------------------------------------%
```

```
let pc_out =
   define_type 'pc_out'
               'pc_out = PCOut busn#busn bool#bool busn#busn busn#busn
                             wire#wire wire#wire bool#bool bool#bool wire#wire
                             wire#wire bool#bool bool#bool';;

let L_ad_out0 = new_recursive_definition
    false
    pc_out
    'L_ad_out0'
    "L_ad_out0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = L_ad_out";;

let L_ready_0 = new_recursive_definition
    false
    pc_out
    'L_ready_0'
    "L_ready_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = L_ready_";;

let I_ad_out0 = new_recursive_definition
    false
    pc_out
    'I_ad_out0'
    "I_ad_out0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_ad_out";;

let I_be_0 = new_recursive_definition
    false
    pc_out
    'I_be_0'
    "I_be_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_be_";;

let I_rale_0 = new_recursive_definition
    false
    pc_out
    'I_rale_0'
    "I_rale_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_rale_";;

let I_male_0 = new_recursive_definition
    false
    pc_out
    'I_male_0'
    "I_male_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_male_";;

let I_crqt_0 = new_recursive_definition
    false
    pc_out
    'I_crqt_0'
    "I_crqt_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_crqt_";;

let I_cale_0 = new_recursive_definition
    false
    pc_out
    'I_cale_0'
    "I_cale_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                      I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_cale_";;

let I_mrdy_0 = new_recursive_definition
    false
```

```
    pc_out
    'I_mrdy_0'
    "I_mrdy_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                   I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_mrdy_";;

let I_last_0 = new_recursive_definition
    false
    pc_out
    'I_last_0'
    "I_last_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                   I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_last_";;

let I_hlda_0 = new_recursive_definition
    false
    pc_out
    'I_hlda_0'
    "I_hlda_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                   I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_hlda_";;

let I_lock_0 = new_recursive_definition
    false
    pc_out
    'I_lock_0'
    "I_lock_0 (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_
                   I_crqt_ I_cale_ I_mrdy_ I_last_ I_hlda_ I_lock_)
     = I_lock_";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm pc_out);;

let Out_Selectors_Work = prove_thm
   ('Out_Selectors_Work',
    "!p:pc_out .
     p = (PCOut (L_ad_out0 p) (L_ready_0 p) (I_ad_out0 p) (I_be_0 p) (I_rale_0 p)
              (I_male_0 p) (I_crqt_0 p) (I_cale_0 p) (I_mrdy_0 p) (I_last_0 p)
              (I_hlda_0 p) (I_lock_0 p))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "p:pc_out" Out_CASES)
    THEN REWRITE_TAC [L_ad_out0; L_ready_0; I_ad_out0; I_be_0; I_rale_0;
                      I_male_0; I_crqt_0; I_cale_0; I_mrdy_0; I_last_0; I_hlda_0;
                      I_lock_0]
   );;

close_theory();;


%-----------------------------------------------------------------------------

   File:          pblock_def.ml

   Author:        (c) D.A. Fura 1992

   Date:          18 February 1993

   This file contains the ml source for the gate-level specification of the PIU
   P-Port, an ASIC developed by the Embedded Processing Laboratory, Boeing High
   Technology Center.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/pport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

set_flag ('timing', true);;

system 'rm pblock_def.th';;
```

```
new_theory 'pblock_def';;

loadf 'aux_defs';;

map new_parent ['paux_def';'wordn_def';'array_def';'ffs_def';'counters_def'];;
map load_parent ['piuaux_def';'gates_def1';'latches_def';'buses_def'];;

%----------------------------------------------------------------------------
P-Port data latches.
----------------------------------------------------------------------------%

let Data_Latches_GATE = new_definition
   ('Data_Latches_GATE',
    "! (lad_in lbe_in_ :time->wordn#wordn) (lwr_in en_in be_sel :time->bool#bool)
       (addr be_ :time->wordn) (dest1 wr :time->bool)
       (data_out addr_out be_out_ :time->wordn#wordn)
       (dest1_out wr_out :time->bool#bool) .
     Data_Latches_GATE lad_in lbe_in_ lwr_in en_in be_sel
                       addr dest1 be_ wr
                       data_out addr_out dest1_out be_out_ wr_out =
     ! t:time .
          (addr (t+1) =
             (ASel(en_in t)) => SUBARRAY (ASel(lad_in t)) (25,0) | addr t) /\
          (dest1 (t+1) =
             (ASel(en_in t)) => ELEMENT (ASel(lad_in t)) (31) | dest1 t) /\
          (be_ (t+1) = (ASel(en_in t)) => ASel(lbe_in_ t) | be_ t) /\
          (wr (t+1) = (ASel(en_in t)) => ASel(lwr_in t) | wr t) /\

          (data_out t = (ASel(lad_in t), ASel(lad_in t))) /\
           (let od1 = MALTER ARBN (31,28) (be_ (t+1)) in
            (let od2 = ALTER od1 (27) (wr (t+1)) in
             (let od3 = ALTER od2 (26) F in
              (let od4 = MALTER od3 (25,24) (SUBARRAY (addr (t+1)) (1,0)) in
               (let od5 = MALTER od4 (23,0) (SUBARRAY (addr (t+1)) (25,2)) in
          (addr_out t = (od5, od5)))))))) /\
          (dest1_out t = ((dest1 (t+1)), (dest1 (t+1)))) /\
          (be_out_ t = (((ASel(be_sel t)) => be_ (t+1) | ASel(lbe_in_ t)) ,
                        ((BSel(be_sel t)) => be_ (t+1) | ASel(lbe_in_ t)))) /\
          (wr_out t = (wr (t+1) , wr (t+1)))"
   );;

%----------------------------------------------------------------------------
Input logic for P_rqt latch.
----------------------------------------------------------------------------%

let Req_Inputs_GATE = new_definition
   ('Req_Inputs_GATE',
    "! (l_ads_ l_den_ reset_rqt :time->bool#bool)
       (rqt_inS rqt_inR rqt_inE :time->bool#bool) .
     Req_Inputs_GATE l_ads_ l_den_ reset_rqt rqt_inS rqt_inR rqt_inE =
        ! t:time .
             (rqt_inS t = ((~ASel(l_ads_ t) /\ ASel(l_den_ t)) ,
                           (~BSel(l_ads_ t) /\ BSel(l_den_ t)))) /\
             (rqt_inR t = (ASel(reset_rqt t) , BSel(reset_rqt t))) /\
             (rqt_inE t = ((ASel(rqt_inS t) \/ ASel(rqt_inR t)) ,
                           (BSel(rqt_inS t) \/ BSel(rqt_inR t))))"
   );;

%----------------------------------------------------------------------------
Input logic for P_size counter.
----------------------------------------------------------------------------%

let Ctr_Logic_GATE = new_definition
('Ctr_Logic_GATE',
 "! (l_ad_in :time->wordn#wordn) (load_in down_in zero_cnt :time->bool#bool)
       (p_size :time->wordn) (p_load p_down :time->bool) .
  Ctr_Logic_GATE l_ad_in load_in down_in zero_cnt p_size p_load p_down =
       ! t:time .
            (p_load (t+1) = BSel(load_in t)) /\
            (p_down (t+1) = BSel(down_in t)) /\
            (p_size (t+1) = (p_load t)
```

```
                                => SUBARRAY (BSel(l_ad_in t)) (1,0) |
  (p_down t) => DECN 1 (p_size t) | p_size t) /\

            (zero_cnt t =
               ((p_size t = (p_down t) => (WORDN 1 1) | (WORDN 1 0)),
                (p_size t = (p_down t) => (WORDN 1 1) | (WORDN 1 0))))"
   );;

%-----------------------------------------------------------------------------
Accumulated random logic.
-----------------------------------------------------------------------------%

let Scat_Logic_GATE = new_definition
   ('Scat_Logic_GATE',
    "! (p_addr :time->wordn#wordn)
       (rst fsm_astate fsm_dstate fsm_hlda_ p_dest1 p_wr p_rqt :time->bool#bool)
       (zero_cnt i_srdy_ i_ad_data_out_en l_ad_out_en_ i_rale_ :time->bool#bool)
       (i_male_ i_crqt_ fsm_mrqt fsm_rst fsm_sack reset_rqt :time->bool#bool)
       (l_ready :time->bool#bool) .
     Scat_Logic_GATE rst fsm_astate fsm_dstate fsm_hlda_ p_addr p_dest1 p_wr
                     p_rqt zero_cnt i_srdy_ i_ad_data_out_en l_ad_out_en_ i_rale_
                     i_male_ i_crqt_ fsm_mrqt fsm_rst fsm_sack reset_rqt
                     l_ready =
        ! t:time .
            (i_ad_data_out_en t = (ASel(p_wr t) /\ ASel(fsm_dstate t)),
                                  (BSel(p_wr t) /\ BSel(fsm_dstate t))) /\
            (l_ad_out_en_ t = ((ASel(fsm_astate t) \/
                                ~ASel(fsm_hlda_ t) \/
                                ASel(fsm_dstate t) /\ ASel(p_wr t)),
                               (BSel(fsm_astate t) \/
                                ~BSel(fsm_hlda_ t) \/
                                BSel(fsm_dstate t) /\ BSel(p_wr t)))) /\
            (i_rale_ t = ((~(~ASel(p_dest1 t) /\
                            ((SUBARRAY (ASel(p_addr t)) (23,22)) = (WORDN 1 3)) /\
                            (ASel(fsm_astate t)) /\
                            (ASel(p_rqt t)))),
                          (~(~BSel(p_dest1 t) /\
                            ((SUBARRAY (BSel(p_addr t)) (23,22)) = (WORDN 1 3)) /\
                            BSel(fsm_astate t) /\
                            BSel(p_rqt t))))) /\
            (i_male_ t = ((~(~ASel(p_dest1 t) /\
                           ~((SUBARRAY (ASel(p_addr t)) (23,22)) = (WORDN 1 3)) /\
                            ASel(fsm_astate t) /\
                            ASel(p_rqt t))),
                          (~(~BSel(p_dest1 t) /\
                           ~((SUBARRAY (BSel(p_addr t)) (23,22)) = (WORDN 1 3)) /\
                            BSel(fsm_astate t) /\
                            BSel(p_rqt t))))) /\
            (i_crqt_ t = ((~(ASel(p_dest1 t) /\
                            (ASel(p_rqt t)))),
                          (~(BSel(p_dest1 t) /\
                           (BSel(p_rqt t))))))  /\
            (fsm_mrqt t = ((~ASel(p_dest1 t) /\ ASel(p_rqt t)),
                           (~BSel(p_dest1 t) /\ BSel(p_rqt t)))) /\
            (fsm_rst t = (ASel(rst t), BSel(rst t))) /\
            (fsm_sack t = ((ASel(zero_cnt t) /\ ~ASel(i_srdy_ t)
                                             /\ ASel(fsm_dstate t)),
                           (BSel(zero_cnt t) /\ ~BSel(i_srdy_ t)
                                             /\ BSel(fsm_dstate t)))) /\
            (reset_rqt t = ((ASel(rst t) \/ ASel(fsm_sack t)),
                            (BSel(rst t) \/ BSel(fsm_sack t)))) /\
            (l_ready t = ((~ASel(i_srdy_ t) /\ ASel(fsm_dstate t)),
                          (~BSel(i_srdy_ t) /\ BSel(fsm_dstate t))))"
   );;

%-----------------------------------------------------------------------------
Input logic for P_lock_ latch.
-----------------------------------------------------------------------------%

let Lock_Inputs_GATE = new_definition
   ('Lock_Inputs_GATE',
    "! (rst fsm_dstate p_male_ p_rale_ :time->bool#bool)
```

```
      (lock_inE lock_inh_inE :time->bool#bool) .
    Lock_Inputs_GATE rst fsm_dstate p_male_ p_rale_ lock_inE lock_inh_inE =
    ! t:time .
      (lock_inE t = (ASel(rst t) \/ ASel(fsm_dstate t)),
                    (BSel(rst t) \/ BSel(fsm_dstate t))) /\
      (lock_inh_inE t = (ASel(rst t) \/ ~ASel(p_male_ t) \/ ~ASel(p_rale_ t)),
                        (BSel(rst t) \/ ~BSel(p_male_ t) \/ ~BSel(p_rale_ t)))"
  );;

%-------------------------------------------------------------------------------
P-Port controller state machine.
-------------------------------------------------------------------------------%

let FSM_GATE = new_definition
   ('FSM_GATE',
    "! (rst_in mrqt_in sack_in cgnt_in_ crqt_in_ hold_in_ :time->bool#bool)
       (lock_in_ :time->bool#bool)
 (state :time->pfsm_ty)
       (rst mrqt sack cgnt_ crqt_ hold_ lock_ :time->bool)
 (astate_out dstate_out hlda_out_ :time->bool#bool) .
  FSM_GATE rst_in mrqt_in sack_in cgnt_in_ crqt_in_ hold_in_ lock_in_
   state rst mrqt sack cgnt_ crqt_ hold_ lock_
  astate_out dstate_out hlda_out_ =
     ! t:time .
       (state (t+1) =
          (rst t) => PA |
          (state t = PH) => ((hold_ t) => PA | PH) |
          (state t = PA) =>
             (((mrqt t) \/
               ((~crqt_ t) /\ ~cgnt_ t)) => PD |
              (((~hold_ t) /\ lock_ t) => PH | PA)) |
          ((sack t /\ hold_ t) => PA |
           (sack t /\ (~hold_ t) /\ ~lock_ t) => PA |
           (sack t /\ (~hold_ t) /\ lock_ t) => PH | PD)) /\
       (rst (t+1) = BSel(rst_in t)) /\
       (mrqt (t+1) = BSel (mrqt_in t)) /\
       (sack (t+1) = BSel(sack_in t)) /\
       (cgnt_ (t+1) = BSel(cgnt_in_ t)) /\
       (crqt_ (t+1) = BSel(crqt_in_ t)) /\
       (hold_ (t+1) = BSel(hold_in_ t)) /\
       (lock_ (t+1) = BSel(lock_in_ t)) /\

       (astate_out t = (state (t+1) = PA) , (state (t+1) = PA)) /\
       (dstate_out t = (state (t+1) = PD) , (state (t+1) = PD)) /\
       (hlda_out_ t = (~(state (t+1) = PH)) , (~(state (t+1) = PH)))"
  );;

%-------------------------------------------------------------------------------
P-Port Block.
-------------------------------------------------------------------------------%

let PBlock_GATE = new_definition
   ('PBlock_GATE',
    "! (s :time->pc_state) (e :time->pc_env) (p :time->pc_out) .
     PBlock_GATE s e p =
        ? (fsm_astate fsm_dstate rqt_ data_out_en reset_rqt :time->bool#bool)
          (data_out addr_out be_out :time->wordn#wordn)
          (ad_data_out ad_addr_out :time->busn#busn)
          (rqt_inS rqt_inR rqt_inE rqt_outQ :time->bool#bool)
          (zero_cnt zero_cnt_ l_ad_out_en_ rale_ male_ :time->bool#bool)
          (fsm_mrqt fsm_rst fsm_sack l_ready i_cgnt lock_inE :time->bool#bool)
          (lock_outQ lock_inh_inE lock_inh_outQ p_male_outQ :time->bool#bool)
          (p_rale_outQ lock_outQ_ dest1_out wr_out :time->bool#bool) .
        (Data_Latches_GATE (sig L_ad_inE e) (sig L_be_E e) (sig L_wrE e) rqt_
                           fsm_astate (sig P_addrS s) (sig P_dest1S s)
                           (sig P_be_S s) (sig P_wrS s) data_out addr_out
                           dest1_out be_out wr_out) /\
        (TRIBUFn_GATE data_out data_out_en ad_data_out) /\
        (TRIBUFn_GATE addr_out fsm_astate ad_addr_out) /\
        (MERGE2n_GATE (31,0) ad_data_out ad_addr_out (sig I_ad_outO p)) /\
        (TRIBUFn_GATE be_out (sig I_hlda_O p) (sig I_be_O p)) /\
        (Req_Inputs_GATE (sig L_ads_E e) (sig L_den_E e) reset_rqt rqt_inS
```

```
                    rqt_inR rqt_inE) /\
        (DSRELatB_GATE GND rqt_inS rqt_inR rqt_inE (sig P_rqtS s) rqt_outQ) /\
        (NOT_GATE rqt_outQ rqt_) /\
        (Ctr_Logic_GATE (sig L_ad_inE e) rqt_ l_ready zero_cnt (sig P_sizeS s)
                        (sig P_loadS s) (sig P_downS s)) /\
        (Scat_Logic_GATE (sig RstE e) fsm_astate fsm_dstate (sig I_hlda_O p)
                         addr_out dest1_out wr_out rqt_outQ zero_cnt
                         (sig I_srdy_E e) data_out_en l_ad_out_en_ rale_ male_
                         (sig I_crqt_O p) fsm_mrqt fsm_rst fsm_sack reset_rqt
                         l_ready) /\
        (TRIBUF_GATE rale_ (sig I_hlda_O p) (sig I_rale_O p)) /\
        (TRIBUF_GATE male_ (sig I_hlda_O p) (sig I_male_O p)) /\
        (TRIBUF_GATE GND (sig I_hlda_O p) (sig I_mrdy_O p)) /\
        (NOT_GATE zero_cnt zero_cnt_) /\
        (TRIBUF_GATE zero_cnt_ (sig I_hlda_O p) (sig I_last_O p)) /\
        (NOT_GATE l_ready (sig L_ready_O p)) /\
        (DSELatB_GATE (sig L_lock_E e) (sig RstE e) lock_inE (sig P_lock_S s)
                      lock_outQ) /\
        (DSELatB_GATE (sig L_lock_E e) (sig RstE e) lock_inh_inE
                      (sig P_lock_inh_S s) lock_inh_outQ) /\
        (Lock_Inputs_GATE (sig RstE e) fsm_dstate p_male_outQ p_rale_outQ
                          lock_inE lock_inh_inE) /\
        (DELatB_GATE male_ fsm_astate (sig P_male_S s) p_male_outQ) /\
        (DELatB_GATE rale_ fsm_astate (sig P_rale_S s) p_rale_outQ) /\
        (NOT_GATE lock_outQ lock_outQ_) /\
        (NAND2_GATE lock_outQ_ lock_inh_outQ (sig I_lock_O p)) /\
        (NOT_GATE (sig I_cgnt_E e) i_cgnt) /\
        (NAND3_GATE i_cgnt fsm_astate (sig I_hold_E e) (sig I_cale_O p)) /\
        (TRINBUFn_GATE (sig I_ad_inE e) l_ad_out_en_ (sig L_ad_outO p)) /\
        (FSM_GATE fsm_rst fsm_mrqt fsm_sack (sig I_cgnt_E e) (sig I_crqt_O p)
                  (sig I_hold_E e) lock_outQ (sig P_fsm_stateS s)
                  (sig P_fsm_rstS s) (sig P_fsm_mrqtS s) (sig P_fsm_sackS s)
                  (sig P_fsm_cgnt_S s) (sig P_fsm_crqt_S s) (sig P_fsm_hold_S s)
                  (sig P_fsm_lock_S s) fsm_astate fsm_dstate (sig I_hlda_O p))"
    );;

let PBlock_EXP = save_thm
    ('PBlock_EXP',
     (BETA_RULE
     (REWRITE_RULE [EXPAND_LET_RULE Data_Latches_GATE;Req_Inputs_GATE;
                    Ctr_Logic_GATE; Scat_Logic_GATE;Lock_Inputs_GATE;FSM_GATE;
                    NOT_GATE;NAND2_GATE;NAND3_GATE;TRIBUF_GATE;
                    TRIBUFn_GATE;TRINBUFn_GATE;DELatB_GATE;DSELatB_GATE;
                    DSRELatB_GATE;MERGE2n_GATE;ASel;BSel;GND;sig]
     (SPEC_ALL PBlock_GATE)))
    );;

close_theory();;


%-----------------------------------------------------------------------------

    File:       pclock_def.ml

    Author:     (c) D.A. Fura 1992

    Date:       18 February 1992

    This file contains the ml source for the clock-level specification of the PIU
    P-Port, an ASIC developed by the Embedded Processing Laboratory, Boeing High
    Technology Center.  The bulk of this code was translated from an M-language
    simulation program using a translator written by P.J. Windley at the
    University of Idaho.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/ftep/piu/hol/pport/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;
```

```
system 'rm pclock_def.th';;

new_theory 'pclock_def';;

loadf 'aux_defs';;

map new_parent ['array_def';'wordn_def';'busn_def'];;
map load_parent ['piuaux_def';'paux_def'];;

new_type_abbrev ('timeC',":num");;

%----------------------------------------------------------------------------
   Next-state definition for P-Port instruction.
----------------------------------------------------------------------------%

let PC_NSF = new_definition
   ('PC_NSF',
    "! (s :pc_state) (e :pc_env) .
     PC_NSF s e =
        let new_P_fsm_state =
           ((P_fsm_rstS s) => PA |
            (P_fsm_stateS s = PH) => ((P_fsm_hold_S s) => PA | PH) |
            (P_fsm_stateS s = PA) =>
               (((P_fsm_mrqtS s) \/
                 ((~P_fsm_crqt_S s) /\ ~P_fsm_cgnt_S s)) => PD |
                (((~P_fsm_hold_S s) /\ P_fsm_lock_S s) => PH | PA)) |
            ((P_fsm_sackS s /\ P_fsm_hold_S s) => PA |
             (P_fsm_sackS s /\ (~P_fsm_hold_S s) /\ ~P_fsm_lock_S s) => PA |
             (P_fsm_sackS s /\ (~P_fsm_hold_S s) /\ P_fsm_lock_S s) => PH | PD))
        in
        let new_P_addr = ((~P_rqtS s)
                          => (SUBARRAY (ASel(L_ad_inE e)) (25,0))
                          | P_addrS s) in
        let new_P_dest1 = ((~P_rqtS s)
                           => (ELEMENT (ASel(L_ad_inE e)) (31))
                           | P_dest1S s) in
        let new_P_be_ = ((~P_rqtS s) => ASel(L_be_E e) | P_be_S s) in
        let new_P_wr = ((~P_rqtS s) => ASel(L_wrE e) | P_wrS s) in
        let new_P_size =
           ((P_loadS s) => (SUBARRAY (BSel(L_ad_inE e)) (1,0)) |
           ((P_downS s) => (DECN 1 (P_sizeS s)) | P_sizeS s)) in
        let p_ale = (~BSel(L_ads_E e) /\ BSel(L_den_E e)) in
        let p_sack =
           ((P_sizeS s = ((P_downS s) => (WORDN 1 1) | (WORDN 1 0))) /\
            ~BSel(I_srdy_E e) /\
            (new_P_fsm_state = PD)) in
        let new_P_rqt  =
           ((p_ale \/ BSel(RstE e) \/ p_sack)
              => ((p_ale /\ ~(BSel(RstE e) \/ p_sack)) => T |
                  (~p_ale /\ (BSel(RstE e) \/ p_sack)) => F |
                  (~p_ale /\ ~(BSel(RstE e) \/ p_sack)) => F | ARB)
            | (P_rqtS s)) in
        let new_P_load = (~new_P_rqt) in
        let new_P_down = (~BSel(I_srdy_E e) /\ (new_P_fsm_state = PD)) in
        let new_P_male_ =
           ((new_P_fsm_state = PA)
              => ~(~new_P_dest1 /\
                   (~((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3))) /\
                   (new_P_fsm_state = PA) /\
                   new_P_rqt)
            | P_male_S s) in
        let new_P_rale_ =
           ((new_P_fsm_state = PA)
              => ~(~new_P_dest1 /\
                   ((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3)) /\
                   (new_P_fsm_state = PA) /\
                   new_P_rqt)
            | P_rale_S s) in
        let new_P_lock_ =
           (((BSel(RstE e)) \/ (new_P_fsm_state = PD))
              => (BSel(RstE e)) => T | BSel(L_lock_E e)
               |  P_lock_S s) in
```

```
        let new_P_lock_inh_ =
           (((BSel(RstE e)) \/ ~new_P_male_ \/ ~new_P_rale_)
               => (BSel(RstE e)) => T | BSel(L_lock_E e)
               |  P_lock_inh_S s) in
        let new_P_fsm_rst = BSel(RstE e) in
        let new_P_fsm_mrqt = (~new_P_dest1 /\ new_P_rqt) in
        let new_P_fsm_sack = p_sack in
        let new_P_fsm_cgnt_ = BSel(I_cgnt_E e) in
        let new_P_fsm_crqt_ = (~(new_P_dest1 /\ new_P_rqt)) in
        let new_P_fsm_hold_ = BSel(I_hold_E e) in
        let new_P_fsm_lock_ = new_P_lock_ in

     (PCState new_P_addr new_P_dest1 new_P_be_ new_P_wr new_P_fsm_state
             new_P_fsm_rst new_P_fsm_mrqt new_P_fsm_sack new_P_fsm_cgnt_
             new_P_fsm_crqt_ new_P_fsm_hold_ new_P_fsm_lock_ new_P_rqt new_P_size
             new_P_load new_P_down new_P_lock_ new_P_lock_inh_ new_P_male_
             new_P_rale_)"
   );;

let PC_NSF_EXP = save_thm
   ('PC_NSF_EXP',
    (EXPAND_LET_RULE (REWRITE_RULE [ASel;BSel] PC_NSF))
   );;

%----------------------------------------------------------------------------
   Output definition for P-Port instruction.
----------------------------------------------------------------------------%

let PC_OF = new_definition
   ('PC_OF',
    "! (s :pc_state) (e :pc_env) .
     PC_OF s e =
        let new_P_fsm_state =
           ((P_fsm_rstS s) => PA |
            (P_fsm_stateS s = PH) => ((P_fsm_hold_S s) => PA | PH) |
            (P_fsm_stateS s = PA) =>
               (((P_fsm_mrqtS s) \/
                 ((~P_fsm_crqt_S s) /\ ~P_fsm_cgnt_S s)) => PD |
                (((~P_fsm_hold_S s) /\ P_fsm_lock_S s) => PH | PA)) |
            ((P_fsm_sackS s /\ P_fsm_hold_S s) => PA |
             (P_fsm_sackS s /\ (~P_fsm_hold_S s) /\ ~P_fsm_lock_S s) => PA |
             (P_fsm_sackS s /\ (~P_fsm_hold_S s) /\ P_fsm_lock_S s) => PH | PD))
        in
        let new_P_addr = ((~P_rqtS s)
                          => (SUBARRAY (ASel(L_ad_inE e)) (25,0))
                          | P_addrS s) in
        let new_P_dest1 = ((~P_rqtS s)
                           => (ELEMENT (ASel(L_ad_inE e)) (31))
                           | P_dest1S s) in
        let new_P_be_ = ((~P_rqtS s) => ASel(L_be_E e) | P_be_S s) in
        let new_P_wr = ((~P_rqtS s) => ASel(L_wrE e) | P_wrS s) in
        let new_P_size =
           ((P_loadS s) => (SUBARRAY (BSel(L_ad_inE e)) (1,0)) |
           ((P_downS s) => (DECN 1 (P_sizeS s)) | P_sizeS s)) in
        let p_ale = (~BSel(L_ads_E e) /\ BSel(L_den_E e)) in
        let p_sack =
           ((P_sizeS s = ((P_downS s) => (WORDN 1 1) | (WORDN 1 0))) /\
            ~BSel(I_srdy_E e) /\
            (new_P_fsm_state = PD)) in
        let new_P_rqt  =
           ((p_ale \/ BSel(RstE e) \/ p_sack)
               => ((p_ale /\ ~(BSel(RstE e) \/ p_sack)) => T |
                   (~p_ale /\ (BSel(RstE e) \/ p_sack)) => F |
                   (~p_ale /\ ~(BSel(RstE e) \/ p_sack)) => F | ARB)
               | (P_rqtS s)) in
        let new_P_load = (~new_P_rqt) in
        let new_P_down = (~BSel(I_srdy_E e) /\ (new_P_fsm_state = PD)) in
        let new_P_male_ =
           ((new_P_fsm_state = PA)
              => ~(~new_P_dest1 /\
                   (~((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3))) /\
                   (new_P_fsm_state = PA) /\
```

```
          new_P_rqt)
      | P_male_S s) in
let new_P_rale_ =
   ((new_P_fsm_state = PA)
      => ~(~new_P_dest1 /\
           ((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3)) /\
           (new_P_fsm_state = PA) /\
           new_P_rqt)
      | P_rale_S s) in
let new_P_lock_ =
   (((BSel(RstE e)) \/ (new_P_fsm_state = PD))
       => (BSel(RstE e)) => T | BSel(L_lock_E e)
       | P_lock_S s) in
let new_P_lock_inh_ =
   (((BSel(RstE e)) \/ ~new_P_male_ \/ ~new_P_rale_)
       => (BSel(RstE e)) => T | BSel(L_lock_E e)
       | P_lock_inh_S s) in
let new_P_fsm_rst = BSel(RstE e) in
let new_P_fsm_mrqt = (~new_P_dest1 /\ new_P_rqt) in
let new_P_fsm_sack = p_sack in
let new_P_fsm_cgnt_ = BSel(I_cgnt_E e) in
let new_P_fsm_crqt_ = (~(new_P_dest1 /\ new_P_rqt)) in
let new_P_fsm_hold_ = BSel(I_hold_E e) in
let new_P_fsm_lock_ = new_P_lock_ in

let lad_en_ =
   ((new_P_fsm_state = PA) \/
    (new_P_fsm_state = PH) \/
    ((new_P_fsm_state = PD) /\ new_P_wr)) in
let L_ad_out =
   (((~lad_en_) => BUSN (ASel(I_ad_inE e)) | Offn),
    ((~lad_en_) => BUSN (BSel(I_ad_inE e)) | Offn)) in
let L_ready_ = ((~(~ASel(I_srdy_E e) /\ (new_P_fsm_state = PD))),
                (~(~BSel(I_srdy_E e) /\ (new_P_fsm_state = PD)))) in
let od0 = ARBN in
let od1 = (MALTER od0 (31,28) new_P_be_) in
let od2 = (ALTER od1 (27) new_P_wr) in
let od3 = (ALTER od2 (26) F) in
let od4 = (MALTER od3 (25,24) (SUBARRAY new_P_addr (1,0))) in
let od5 = (MALTER od4 (23,0) (SUBARRAY new_P_addr (25,2))) in
let I_ad_out =
  (((new_P_wr /\ (new_P_fsm_state = PD)) => BUSN (ASel(L_ad_inE e)) |
    (new_P_fsm_state = PA) => BUSN od5 | Offn),
   ((new_P_wr /\ (new_P_fsm_state = PD)) => BUSN (ASel(L_ad_inE e)) |
    (new_P_fsm_state = PA) => BUSN od5 | Offn)) in
let I_be_ =
   (((~(new_P_fsm_state = PH))
       => BUSN ((new_P_fsm_state = PA) => new_P_be_ | ASel(L_be_E e))
       | Offn),
    ((~(new_P_fsm_state = PH))
       => BUSN ((new_P_fsm_state = PA) => new_P_be_ | ASel(L_be_E e))
       | Offn)) in
let rale_outA_ = (~(~new_P_dest1 /\
                    ((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3)) /\
                    (new_P_fsm_state = PA) /\
                    (P_rqtS s))) in
let rale_outB_ = (~(~new_P_dest1 /\
                    ((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3)) /\
                    (new_P_fsm_state = PA) /\
                    new_P_rqt)) in
let I_rale_ = (((~(new_P_fsm_state = PH)) => WIRE rale_outA_ | Z),
               ((~(new_P_fsm_state = PH)) => WIRE rale_outB_ | Z)) in
let male_outA_ = (~(~new_P_dest1 /\
                    (~((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3))) /\
                    (new_P_fsm_state = PA) /\
                    (P_rqtS s))) in
let male_outB_ = (~(~new_P_dest1 /\
                    (~((SUBARRAY new_P_addr (25,24)) = (WORDN 1 3))) /\
                    (new_P_fsm_state = PA) /\
                    new_P_rqt)) in
let I_male_ = (((~(new_P_fsm_state = PH)) => WIRE male_outA_ | Z),
               ((~(new_P_fsm_state = PH)) => WIRE male_outB_ | Z)) in
```

```
        let I_crqt_ = ((~(new_P_dest1 /\ (P_rqtS s))),
                       (~(new_P_dest1 /\ new_P_rqt))) in
        let I_cale_ = ((~(~ASel(I_cgnt_E e) /\
                          (new_P_fsm_state = PA) /\
                          ASel(I_hold_E e))),
                       (~(~BSel(I_cgnt_E e) /\
                          (new_P_fsm_state = PA) /\
                          BSel(I_hold_E e)))) in
        let I_mrdy_ = ((((~(new_P_fsm_state = PH)) => LO | Z)),
                       (((~(new_P_fsm_state = PH)) => LO | Z))) in
        let last_out_ =
           (~(P_sizeS s = ((P_downS s) => (WORDN 1 1) | (WORDN 1 0)))) in
        let I_last_ = ((((~(new_P_fsm_state = PH)) => WIRE last_out_ | Z)),
                       (((~(new_P_fsm_state = PH)) => WIRE last_out_ | Z))) in
        let I_hlda_ = ((~(new_P_fsm_state = PH)) , (~(new_P_fsm_state = PH))) in
        let I_lock_ = ((~(~(P_lock_S s) /\ (P_lock_inh_S s))),
                       (~(~new_P_lock_ /\ new_P_lock_inh_))) in

        (PCOut L_ad_out L_ready_ I_ad_out I_be_ I_rale_ I_male_ I_crqt_ I_cale_
              I_mrdy_ I_last_ I_hlda_ I_lock_)"
   );;

let PC_OF_EXP = save_thm
   ('PC_OF_EXP',
    (EXPAND_LET_RULE (REWRITE_RULE [ASel;BSel] PC_OF))
   );;

let PC_Exec = new_definition
   ('PC_Exec',
    "! (pci :PCI) (s :timeC->pc_state) (e :timeC->pc_env) (p :timeC->pc_out)
       (t :timeC) .
     PC_Exec pci s e p t = T"
   );;

let PC_PreC = new_definition
   ('PC_PreC',
    "! (pci :PCI) (s :timeC->pc_state) (e :timeC->pc_env) (p :timeC->pc_out)
       (t :timeC) .
     PC_PreC pci s e p t = T"
   );;

let PC_PostC = new_definition
   ('PC_PostC',
    "! (pci :PCI) (s :timeC->pc_state) (e :timeC->pc_env) (p :timeC->pc_out)
       (t :timeC) .
     PC_PostC pci s e p t =
        (s (t+1) = PC_NSF (s t) (e t)) /\
        (p t = PC_OF (s t) (e t))"
   );;

let PC_Correct = new_definition
   ('PC_Correct',
    "! (pci :PCI) (s :timeC->pc_state) (e :timeC->pc_env) (p :timeC->pc_out)
       (t :timeC) .
     PC_Correct pci s e p t =
        PC_Exec pci s e p t /\
        PC_PreC pci s e p t
        ==>
        PC_PostC pci s e p t"
   );;

let PCSet_Correct = new_definition
   ('PCSet_Correct',
    "! (s :timeC->pc_state) (e :timeC->pc_env) (p :timeC->pc_out) .
     PCSet_Correct s e p = !(pci:PCI)(t:timeC). PC_Correct pci s e p t"
   );;

let Next_State_THM = TAC_PROOF
   (([], "PCSet_Correct s e p ==> (s (t+1) = PC_NSF (s t) (e t))"),
    REWRITE_TAC [PCSet_Correct;PC_Correct;PC_Exec;PC_PreC;PC_PostC]
    THEN REWRITE_TAC
       [LEFT_IMP_FORALL_CONV
```

```
        "(!t.
          (s(t + 1) = PC_NSF(s t)(e t)) /\
          (p t = PC_OF(s t)(e t))) ==>
          (s (t+1) = PC_NSF(s t)(e t))"]
    THEN EXISTS_TAC "t:time"
    THEN REWRITE_TAC [ADD_CLAUSES]
    THEN STRIP_TAC
   );;

let P_addr_ISO = save_thm
   ('P_addr_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_addrS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_addrS (s ((t:timeC) + 1))"))))
   );;

let P_dest1_ISO = save_thm
   ('P_dest1_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_dest1S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_dest1S (s ((t:timeC) + 1))"))))
   );;

let P_be_ISO = save_thm
   ('P_be_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_be_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_be_S (s ((t:timeC) + 1))"))))
   );;

let P_wr_ISO = save_thm
   ('P_wr_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_wrS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_wrS (s ((t:timeC) + 1))"))))
   );;

let P_fsm_state_ISO = save_thm
   ('P_fsm_state_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_stateS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_stateS (s ((t:timeC) + 1))"))))
   );;

let P_fsm_rst_ISO = save_thm
   ('P_fsm_rst_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_rstS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_rstS (s ((t:timeC) + 1))"))))
   );;

let P_fsm_mrqt_ISO = save_thm
   ('P_fsm_mrqt_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_mrqtS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_mrqtS (s ((t:timeC) + 1))"))))
   );;
```

```
let P_fsm_sack_ISO = save_thm
   ('P_fsm_sack_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_sackS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_sackS (s ((t:timeC) + 1))"))))
   );;

 let P_fsm_cgnt_ISO = save_thm
   ('P_fsm_cgnt_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_cgnt_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_cgnt_S (s ((t:timeC) + 1))"))))
   );;

let P_fsm_crqt_ISO = save_thm
   ('P_fsm_crqt_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_crqt_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_crqt_S (s ((t:timeC) + 1))"))))
   );;

let P_fsm_hold_ISO = save_thm
   ('P_fsm_hold_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_hold_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_hold_S (s ((t:timeC) + 1))"))))
   );;

let P_fsm_lock_ISO = save_thm
   ('P_fsm_lock_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_fsm_lock_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_fsm_lock_S (s ((t:timeC) + 1))"))))
   );;

let P_rqt_ISO = save_thm
   ('P_rqt_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_rqtS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_rqtS (s ((t:timeC) + 1))"))))
   );;

let P_size_ISO = save_thm
   ('P_size_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_sizeS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_sizeS (s ((t:timeC) + 1))"))))
   );;

let P_load_ISO = save_thm
   ('P_load_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_loadS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_loadS (s ((t:timeC) + 1))"))))
   );;
```

```
let P_down_ISO = save_thm
   ('P_down_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_downS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_downS (s ((t:timeC) + 1))"))))
   );;

let P_lock_ISO = save_thm
   ('P_lock_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_lock_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_lock_S (s ((t:timeC) + 1))"))))
   );;

let P_lock_inh_ISO = save_thm
   ('P_lock_inh_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_lock_inh_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_lock_inh_S (s ((t:timeC) + 1))"))))
   );;

let P_male_ISO = save_thm
   ('P_male_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_male_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_male_S (s ((t:timeC) + 1))"))))
   );;

let P_rale_ISO = save_thm
   ('P_rale_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_NSF_EXP;P_rale_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "P_rale_S (s ((t:timeC) + 1))"))))
   );;

let Out_THM = TAC_PROOF
   (([], "PCSet_Correct s e p ==> (p t = PC_OF (s t) (e t))"),
    REWRITE_TAC [PCSet_Correct;PC_Correct;PC_Exec;PC_PreC;PC_PostC]
    THEN REWRITE_TAC
       [LEFT_IMP_FORALL_CONV
          "(!t.
            (s (t + 1) = PC_NSF(s t)(e t)) /\
            (p t = PC_OF(s t)(e t))) ==>
            (p t = PC_OF(s t)(e t))"]
    THEN EXISTS_TAC "t:time"
    THEN REWRITE_TAC [ADD_CLAUSES]
    THEN STRIP_TAC
   );;

let L_ad_out_ISO = save_thm
   ('L_ad_out_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_OF_EXP;L_ad_outO]
       (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "L_ad_outO (p (t:timeC))"))))
   );;

let L_ready_ISO = save_thm
   ('L_ready_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_OF_EXP;L_ready_O]
```

```
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "L_ready_O (p (t:timeC))"))))
  );;

let I_ad_out_ISO = save_thm
  ('I_ad_out_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_ad_outO]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_ad_outO (p (t:timeC))")))));
  );;

let I_be_ISO = save_thm
  ('I_be_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_be_O]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_be_O (p (t:timeC))"))))
  );;

let I_rale_ISO = save_thm
  ('I_rale_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_rale_O]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_rale_O (p (t:timeC))"))))
  );;

let I_male_ISO = save_thm
  ('I_male_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_male_O]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_male_O (p (t:timeC))"))))
  );;

let I_crqt_ISO = save_thm
  ('I_crqt_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_crqt_O]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_crqt_O (p (t:timeC))"))))
  );;

let I_cale_ISO = save_thm
  ('I_cale_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_cale_O]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_cale_O (p (t:timeC))"))))
  );;

let I_mrdy_ISO = save_thm
  ('I_mrdy_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_mrdy_O]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_mrdy_O (p (t:timeC))"))))
  );;

let I_last_ISO = save_thm
  ('I_last_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_last_O]
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_last_O (p (t:timeC))"))))
  );;

let I_hlda_ISO = save_thm
  ('I_hlda_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [PC_OF_EXP;I_hlda_O]
```

```
      (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_hlda_0 (p (t:timeC))"))))
   );;

let I_lock_ISO = save_thm
   ('I_lock_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [PC_OF_EXP;I_lock_0]
       (SUBS_OCCS [([2],UNDISCH (Out_THM))] (REFL "I_lock_0 (p (t:timeC))")))))
   );;

close_theory();;
```

3.3 M-Port Definitions

This section contains the theories *maux_def*, *mblock_def*, and *mclock_def*, defining the M-Port design.

```
%-----------------------------------------------------------------------------

   File:        maux_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        15 January 1993

-----------------------------------------------------------------------------%

set_flag ('timing', true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm maux_def.th';;

new_theory 'maux_def';;

map new_parent ['busn_def';'less_eq'];;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('wordn', ":(num->bool)");;
new_type_abbrev ('busn', ":(num->wire)");;

%---------------------------------------------------------------------------
   Abstract data type for the M-Port FSM states.
---------------------------------------------------------------------------%

let mfsm_ty_Axiom =
   define_type 'mfsm_ty_Axiom'
               'mfsm_ty = MI | MA | MW | MRR | MR | MBW';;

%---------------------------------------------------------------------------
   Abstract data type for the M-Port instruction.
---------------------------------------------------------------------------%

let MCI =
   define_type 'MCI'
               'MCI = MC_X';;

%---------------------------------------------------------------------------
   Abstract data type for the state.
---------------------------------------------------------------------------%

let mc_state =
   define_type 'mc_state'
               'mc_state = MCState mfsm_ty bool bool bool bool bool bool bool bool
                           bool bool wordn wordn wordn bool bool bool
```

```
                                wordn wordn';;

let M_fsm_stateS = new_recursive_definition
    false
    mc_state
    'M_fsm_stateS'
    "M_fsm_stateS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_state";;

let M_fsm_male_S = new_recursive_definition
    false
    mc_state
    'M_fsm_male_S'
    "M_fsm_male_S (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_male_";;

let M_fsm_rdS = new_recursive_definition
    false
    mc_state
    'M_fsm_rdS'
    "M_fsm_rdS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_rd";;

let M_fsm_bwS = new_recursive_definition
    false
    mc_state
    'M_fsm_bwS'
    "M_fsm_bwS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_bw";;

let M_fsm_wwS = new_recursive_definition
    false
    mc_state
    'M_fsm_wwS'
    "M_fsm_wwS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_ww";;

let M_fsm_last_S = new_recursive_definition
    false
    mc_state
    'M_fsm_last_S'
    "M_fsm_last_S (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_last_";;

let M_fsm_mrdy_S = new_recursive_definition
    false
    mc_state
    'M_fsm_mrdy_S'
    "M_fsm_mrdy_S (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_mrdy_";;

let M_fsm_zero_cntS = new_recursive_definition
    false
    mc_state
    'M_fsm_zero_cntS'
    "M_fsm_zero_cntS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
```

```
     = M_fsm_zero_cnt";;

let M_fsm_rstS = new_recursive_definition
    false
    mc_state
    'M_fsm_rstS'
    "M_fsm_rstS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_fsm_rst";;

let M_seS = new_recursive_definition
    false
    mc_state
    'M_seS'
    "M_seS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_se";;

let M_wrS = new_recursive_definition
    false
    mc_state
    'M_wrS'
    "M_wrS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_wr";;

let M_addrS = new_recursive_definition
    false
    mc_state
    'M_addrS'
    "M_addrS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_addr";;

let M_beS = new_recursive_definition
    false
    mc_state
    'M_beS'
    "M_beS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_be";;

let M_countS = new_recursive_definition
    false
    mc_state
    'M_countS'
    "M_countS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_count";;

let M_rdyS = new_recursive_definition
    false
    mc_state
    'M_rdyS'
    "M_rdyS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_rdy";;

let M_wwdelS = new_recursive_definition
    false
    mc_state
    'M_wwdelS'
    "M_wwdelS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                 M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                 M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
```

```
     = M_wwdel";;

let M_parityS = new_recursive_definition
    false
    mc_state
    'M_parityS'
    "M_parityS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_parity";;

let M_rd_dataS = new_recursive_definition
    false
    mc_state
    'M_rd_dataS'
    "M_rd_dataS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_rd_data";;

let M_detectS = new_recursive_definition
    false
    mc_state
    'M_detectS'
    "M_detectS (MCState M_fsm_state M_fsm_male_ M_fsm_rd M_fsm_bw M_fsm_ww
                   M_fsm_last_ M_fsm_mrdy_ M_fsm_zero_cnt M_fsm_rst M_se M_wr
                   M_addr M_be M_count M_rdy M_wwdel M_parity M_rd_data M_detect)
     = M_detect";;

let State_CASES =
   prove_cases_thm (prove_induction_thm mc_state);;

let State_Selectors_Work = prove_thm
   ('State_Selectors_Work',
    "! s:mc_state .
     s = (MCState (M_fsm_stateS s) (M_fsm_male_S s) (M_fsm_rdS s) (M_fsm_bwS s)
                (M_fsm_wwS s) (M_fsm_last_S s) (M_fsm_mrdy_S s)
                (M_fsm_zero_cntS s) (M_fsm_rstS s) (M_seS s) (M_wrS s)
                (M_addrS s) (M_beS s) (M_countS s) (M_rdyS s) (M_wwdelS s)
                (M_parityS s) (M_rd_dataS s) (M_detectS s))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "s:mc_state" State_CASES)
     THEN REWRITE_TAC [M_fsm_stateS; M_fsm_male_S; M_fsm_rdS; M_fsm_bwS;
                       M_fsm_wwS; M_fsm_last_S; M_fsm_mrdy_S; M_fsm_zero_cntS;
                       M_fsm_rstS; M_seS; M_wrS; M_addrS; M_beS; M_countS;
                       M_rdyS; M_wwdelS; M_parityS; M_rd_dataS; M_detectS]
   );;

%----------------------------------------------------------------------------
   Abstract data type for the environment.
----------------------------------------------------------------------------%

let mc_env =
   define_type 'mc_env'
               'mc_env = MCEnv bool#bool bool#bool bool#bool wordn#wordn
                             bool#bool bool#bool bool#bool bool#bool
                                        bool#bool  wordn#wordn  bool#bool  wordn#wordn  bool#bool
bool#bool';;

let RstE = new_recursive_definition
    false
    mc_env
    'RstE'
    "RstE (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
               I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
               Edac_en_ Reset_parity)
     = Rst";;

let Disable_eepromE = new_recursive_definition
    false
    mc_env
    'Disable_eepromE'
```

```
    "Disable_eepromE (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = Disable_eeprom";;

let Disable_writesE = new_recursive_definition
    false
    mc_env
    'Disable_writesE'
    "Disable_writesE (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = Disable_writes";;

let I_ad_inE = new_recursive_definition
    false
    mc_env
    'I_ad_inE'
    "I_ad_inE (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_ad_in";;

let I_male_E = new_recursive_definition
    false
    mc_env
    'I_male_E'
    "I_male_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_male_";;

let I_rale_E = new_recursive_definition
    false
    mc_env
    'I_rale_E'
    "I_rale_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_rale_";;

let I_cale_E = new_recursive_definition
    false
    mc_env
    'I_cale_E'
    "I_cale_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_cale_";;

let I_hlda_E = new_recursive_definition
    false
    mc_env
    'I_hlda_E'
    "I_hlda_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_hlda_";;

let I_last_E = new_recursive_definition
    false
    mc_env
    'I_last_E'
    "I_last_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_last_";;

let I_be_E = new_recursive_definition
    false
    mc_env
    'I_be_E'
```

```
    "I_be_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_be_";;

let I_mrdy_E = new_recursive_definition
    false
    mc_env
    'I_mrdy_E'
    "I_mrdy_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = I_mrdy_";;

let MB_data_inE = new_recursive_definition
    false
    mc_env
    'MB_data_inE'
    "MB_data_inE (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = MB_data_in";;

let Edac_en_E = new_recursive_definition
    false
    mc_env
    'Edac_en_E'
    "Edac_en_E (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = Edac_en_";;

let Reset_parityE = new_recursive_definition
    false
    mc_env
    'Reset_parityE'
    "Reset_parityE (MCEnv Rst Disable_eeprom Disable_writes I_ad_in I_male_
              I_rale_ I_cale_ I_hlda_ I_last_ I_be_ I_mrdy_ MB_data_in
              Edac_en_ Reset_parity)
     = Reset_parity";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm mc_env);;

let Env_Selectors_Work = prove_thm
   ('Env_Selectors_Work',
    "! e:mc_env .
     e = (MCEnv (RstE e) (Disable_eepromE e) (Disable_writesE e) (I_ad_inE e)
              (I_male_E e) (I_rale_E e) (I_cale_E e) (I_hlda_E e) (I_last_E e)
              (I_be_E e) (I_mrdy_E e) (MB_data_inE e)
              (Edac_en_E e) (Reset_parityE e))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "e:mc_env" Env_CASES)
     THEN REWRITE_TAC [RstE; Disable_eepromE; Disable_writesE; I_ad_inE;
                       I_male_E; I_rale_E; I_cale_E; I_hlda_E; I_last_E; I_be_E;
                       I_mrdy_E; MB_data_inE; Edac_en_E; Reset_parityE]
   );;

%----------------------------------------------------------------------------
   Abstract data type for the output.
----------------------------------------------------------------------------%

let mc_out =
   define_type 'mc_out'
               'mc_out = MCOut busn#busn wire#wire wordn#wordn busn#busn
                         bool#bool bool#bool bool#bool bool#bool bool#bool';;

let I_ad_outO = new_recursive_definition
    false
    mc_out
    'I_ad_outO'
    "I_ad_outO (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
```

```
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = I_ad_out";;

let I_srdy_O = new_recursive_definition
    false
    mc_out
    'I_srdy_O'
    "I_srdy_O (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = I_srdy_";;

let MB_addrO = new_recursive_definition
    false
    mc_out
    'MB_addrO'
    "MB_addrO (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = MB_addr";;

let MB_data_outO = new_recursive_definition
    false
    mc_out
    'MB_data_outO'
    "MB_data_outO (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = MB_data_out";;

let MB_cs_eeprom_O = new_recursive_definition
    false
    mc_out
    'MB_cs_eeprom_O'
    "MB_cs_eeprom_O (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = MB_cs_eeprom_";;

let MB_cs_sram_O = new_recursive_definition
    false
    mc_out
    'MB_cs_sram_O'
    "MB_cs_sram_O (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = MB_cs_sram_";;

let MB_we_O = new_recursive_definition
    false
    mc_out
    'MB_we_O'
    "MB_we_O (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = MB_we_";;

let MB_oe_O = new_recursive_definition
    false
    mc_out
    'MB_oe_O'
    "MB_oe_O (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = MB_oe_";;

let MB_parityO = new_recursive_definition
    false
    mc_out
    'MB_parityO'
    "MB_parityO (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_
                    MB_cs_sram_ MB_we_ MB_oe_ MB_parity)
     = MB_parity";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm mc_out);;

let Out_Selectors_Work = prove_thm
   ('Out_Selectors_Work',
```

```
    "! p:mc_out .
     p = (MCOut (I_ad_out0 p) (I_srdy_0 p) (MB_addr0 p) (MB_data_out0 p)
              (MB_cs_eeprom_0 p) (MB_cs_sram_0 p) (MB_we_0 p) (MB_oe_0 p)
              (MB_parity0 p))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "p:mc_out" Out_CASES)
     THEN REWRITE_TAC [I_ad_out0; I_srdy_0; MB_addr0; MB_data_out0;
                       MB_cs_eeprom_0; MB_cs_sram_0; MB_we_0; MB_oe_0;
                       MB_parity0]
   );;

close_theory();;


%-----------------------------------------------------------------------------

   File:        m_block.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        1 March 1993

   This file contains the ml source for the gate-level specification of the
   M-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/mport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/ml/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing', true);;

system 'rm mblock_def.th';;

new_theory 'mblock_def';;

loadf 'abs_theory';;
loadf 'aux_defs';;

map new_parent ['maux_def';'wordn_def';'array_def';'less_eq'];;
map load_parent ['piuaux_def';'gates_def1';'latches_def';'ffs_def'];;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%-----------------------------------------------------------------------------
SRAM/EEPROM selection logic.
-----------------------------------------------------------------------------%

let SE_Logic_GATE = new_definition
   ('SE_Logic_GATE',
    "! (i_ad :time->wordn#wordn)
       (male mem_enable cs_e_ cs_s_ :time->bool#bool)
       (M_se :time->bool) .
     SE_Logic_GATE i_ad male mem_enable M_se cs_e_ cs_s_ =
        ! t:time .
        (M_se (t+1) =
           (BSel(male t)) => (ELEMENT (BSel(i_ad t)) (23)) | M_se t) /\
        (cs_e_ t = ((~ASel(mem_enable t) \/ M_se t),
                    (~BSel(mem_enable t) \/ M_se (t+1)))) /\
        (cs_s_ t = ((~ASel(mem_enable t) \/ ~M_se t),
                    (~BSel(mem_enable t) \/ ~M_se (t+1))))"
   );;

%-----------------------------------------------------------------------------
Read/write selection logic.
-----------------------------------------------------------------------------%
```

```
let WR_Logic_GATE = new_definition
   ('WR_Logic_GATE',
    "! (i_ad :time->wordn#wordn)
       (male mem_enable wr rd_mem wr_mem :time->bool#bool)
       (M_wr :time->bool) .
     WR_Logic_GATE i_ad male mem_enable M_wr wr rd_mem wr_mem =
        ! t:time .
        (M_wr (t+1) =
           (BSel(male t)) => (ELEMENT (BSel(i_ad t)) (27)) | M_wr t) /\
        (wr t = (M_wr t, M_wr (t+1))) /\
        (rd_mem t = ((ASel(mem_enable t) /\ ~M_wr t),
                     (BSel(mem_enable t) /\ ~M_wr (t+1)))) /\
        (wr_mem t = ((ASel(mem_enable t) /\ M_wr t),
                     (BSel(mem_enable t) /\ M_wr (t+1))))"
   );;

%-------------------------------------------------------------------------------
Address counter logic.
-------------------------------------------------------------------------------%

let Addr_Ctr_GATE = new_definition
   ('Addr_Ctr_GATE',
    "! (i_ad addr_out : time->wordn#wordn)
       (male rdy :time->bool#bool)
       (M_addr :time->wordn) .
     Addr_Ctr_GATE i_ad male rdy M_addr addr_out =
        ! t:time .
        (M_addr (t+1) =
           (BSel(male t)) => (SUBARRAY (BSel(i_ad t)) (18,0)) |
           (BSel(rdy t)) => (INCN 18 (M_addr t)) | M_addr t) /\
        (addr_out t =
           (((ASel(rdy t)) => (INCN 18 (M_addr t)) | M_addr t),
            ((BSel(rdy t)) => (INCN 18 (M_addr t)) | M_addr t)))"
   );;

%-------------------------------------------------------------------------------
Byte enable logic.
-------------------------------------------------------------------------------%

let BE_Logic_GATE = new_definition
   ('BE_Logic_GATE',
    "! (i_be_ be_out :time->wordn#wordn)
       (male srdy wr_mem ww bw :time->bool#bool)
       (M_be :time->wordn) .
     BE_Logic_GATE i_be_ male srdy wr_mem M_be be_out ww bw =
        ! t:time .
        (M_be (t+1) =
           (BSel(male t) \/ BSel(srdy t))
              => (NOTN 3 (BSel(i_be_ t))) | (M_be t)) /\
        (be_out t = (M_be t, M_be t)) /\
        (ww t = ((ASel(wr_mem t) /\ (M_be t = (WORDN 3 15))),
                 (BSel(wr_mem t) /\ (M_be (t+1) = (WORDN 3 15))))) /\
        (bw t = ((ASel(wr_mem t) /\ ~(M_be t = (WORDN 3 15))),
                 (BSel(wr_mem t) /\ ~(M_be (t+1) = (WORDN 3 15)))))"
   );;

%-------------------------------------------------------------------------------
Input logic for M_rdy latch.
-------------------------------------------------------------------------------%

let Rdy_Logic_GATE = new_definition
   ('Rdy_Logic_GATE',
    "! (write read zero_cnt wr_mem rdy :time->bool#bool) .
     Rdy_Logic_GATE write read zero_cnt wr_mem rdy =
       ! t:time .
       rdy t = (((ASel(write t) /\ ASel(zero_cnt t)) \/
                 (ASel(read t) /\ ASel(zero_cnt t) /\ ~ASel(wr_mem t))),
                ((BSel(write t) /\ BSel(zero_cnt t)) \/
                 (BSel(read t) /\ BSel(zero_cnt t) /\ ~BSel(wr_mem t))))"
   );;
```

```
%-----------------------------------------------------------------------------
Wait state counter logic.
-----------------------------------------------------------------------------%

let Ctr_Logic_GATE = new_definition
   ('Ctr_Logic_GATE',
    "! (dn ld in zero_cnt :time->bool#bool)
       (M_count :time->wordn) .
     Ctr_Logic_GATE in dn ld M_count zero_cnt =
        ! t:time .
        (M_count (t+1) =
          BSel(ld t) => (BSel(in t) => (WORDN 1 1) | (WORDN 1 2)) |
          BSel(dn t) => (DECN 1 (M_count t)) | (M_count t)) /\
        (zero_cnt t = ((M_count t = (ASel(dn t) => (WORDN 1 1) | (WORDN 1 0))),
                       (M_count t = (BSel(dn t) => (WORDN 1 1) | (WORDN 1 0)))))"
   );;

%-----------------------------------------------------------------------------
Memory control signal logic.
-----------------------------------------------------------------------------%

let Enable_Logic_GATE = new_definition
   ('Enable_Logic_GATE',
    "! (cs_eeprom_ rd_mem address read write byte_write wwdel :time->bool#bool)
       (disable_eeprom disable_writes oe_ edac_le we_ :time->bool#bool)
       (mb_wr_en_ :time->bool#bool) .
     Enable_Logic_GATE cs_eeprom_ rd_mem address read write byte_write wwdel
                       disable_eeprom disable_writes oe_ edac_le we_ mb_wr_en_ =
       ! t:time .
       (oe_ t = ((~((ASel(rd_mem t) /\ ASel(address t)) \/ ASel(read t))),
                 (~((BSel(rd_mem t) /\ BSel(address t)) \/ BSel(read t))))) /\
       (we_ t = ((~((ASel(cs_eeprom_ t) \/ ~ASel(disable_eeprom t)) /\
                    ~ASel(disable_writes t) /\
                    (ASel(byte_write t) \/ ASel(write t) \/ ASel(wwdel t)))),
                 (~((BSel(cs_eeprom_ t) \/ ~BSel(disable_eeprom t)) /\
                    ~BSel(disable_writes t) /\
                    (BSel(byte_write t) \/ BSel(write t) \/ BSel(wwdel t)))))) /\
       (edac_le t = (ASel(read t), BSel(read t))) /\
       (mb_wr_en_ t = ((~ASel(write t)), ~BSel(write t)))"
   );;

%-----------------------------------------------------------------------------
Generation logic for I_srdy_.
-----------------------------------------------------------------------------%

let Srdy_Logic_GATE = new_definition
   ('Srdy_Logic_GATE',
    "! (wr rdy rdy_outQ srdy_ :time->bool#bool) .
     Srdy_Logic_GATE wr rdy rdy_outQ srdy_ =
        ! t:time .
        srdy_ t = ((~((ASel(rdy_outQ t) /\ ~ASel(wr t)) \/
                      (ASel(rdy t) /\ ASel(wr t)))),
                  (~((BSel(rdy_outQ t) /\ ~BSel(wr t)) \/
                     (BSel(rdy t) /\ BSel(wr t)))))"
   );;

%-----------------------------------------------------------------------------
Memory decode logic.
-----------------------------------------------------------------------------%

let EDAC_Decode_Logic_GATE = new_definition
   ('EDAC_Decode_Logic_GATE',
    "! (rep :^REP_ty)
       (mb_data_in data_out detect_out :time->wordn#wordn)
       (edac_en :time->bool#bool) .
     EDAC_Decode_Logic_GATE rep mb_data_in edac_en data_out detect_out =
        ! t:time .
        (data_out t =
           ((ASel(edac_en t) => (Ham_Dec rep (ASel(mb_data_in t)))
                              | ASel(mb_data_in t)),
            (BSel(edac_en t) => (Ham_Dec rep (BSel(mb_data_in t)))
                              | BSel(mb_data_in t)))) /\
```

```
        (detect_out t =
           ((ASel(edac_en t) => (Ham_Det1 rep (ASel(mb_data_in t)))
                             |  (WORDN 3 0)),
            (BSel(edac_en t) => (Ham_Det1 rep (BSel(mb_data_in t)))
                             |  (WORDN 3 0))))"
   );;

%-----------------------------------------------------------------------------
Memory read latches.
-----------------------------------------------------------------------------%

let Read_Latches_GATE = new_definition
   ('Read_Latches_GATE',
    "! (rep:^REP_ty)
       (data_inD detect_inD m_data_outQ :time->wordn#wordn)
       (edac_en edac_le detect_inE m_detect_outQ :time->bool#bool)
       (M_rd_data M_detect :time->wordn) .
     Read_Latches_GATE rep data_inD edac_en edac_le detect_inD detect_inE
                       M_rd_data M_detect m_data_outQ m_detect_outQ =
         ! t:time .
         (M_rd_data (t+1) =
            BSel(edac_le t) => (BSel(data_inD t)) | (M_rd_data t)) /\
         (M_detect (t+1) =
            BSel(detect_inE t) => (BSel(detect_inD t)) | (M_detect t)) /\
         (m_data_outQ t = (M_rd_data t, M_rd_data t)) /\
         (m_detect_outQ t =
            ((Ham_Det2 rep (M_detect t, ASel(edac_en t))),
             (Ham_Det2 rep (M_detect (t+1), BSel(edac_en t)))))"
   );;

%-----------------------------------------------------------------------------
Enable input logic for EDAC correction reporting.
-----------------------------------------------------------------------------%

let Detect_Enable_Logic_GATE = new_definition
   ('Detect_Enable_Logic_GATE',
    "! (edac_en rd_mem detect_inE :time->bool#bool) .
     Detect_Enable_Logic_GATE edac_en rd_mem detect_inE =
         ! t:time .
         detect_inE t =
           (((ASel(edac_en t) /\ ASel(rd_mem t)) \/ ~ASel(rd_mem t)),
            ((BSel(edac_en t) /\ BSel(rd_mem t)) \/ ~BSel(rd_mem t)))"
   );;

%-----------------------------------------------------------------------------
Memory write data multiplexer.
-----------------------------------------------------------------------------%

let Mux_Out_Logic_GATE = new_definition
   ('Mux_Out_Logic_GATE',
    "! (m_data_outQ i_ad be mb_data_out :time->wordn#wordn) .
     Mux_Out_Logic_GATE m_data_outQ i_ad be mb_data_out =
         ! t:time .
         let od1A =
            (MALTER
               ARBN
               (7,0)
               ((ELEMENT (ASel(be t)) (0))
                    => (SUBARRAY (ASel(i_ad t)) (7,0))
                    |  (SUBARRAY (ASel(m_data_outQ t)) (7,0)))) in
         let od2A =
            (MALTER
               od1A
               (15,8)
               ((ELEMENT (ASel(be t)) (1))
                    => (SUBARRAY (ASel(i_ad t)) (15,8))
                    |  (SUBARRAY (ASel(m_data_outQ t)) (15,8)))) in
         let od3A =
            (MALTER
               od2A
               (23,16)
               ((ELEMENT (ASel(be t)) (2))
```

```
                => (SUBARRAY (ASel(i_ad t)) (23,16))
                |  (SUBARRAY (ASel(m_data_outQ t)) (23,16)))) in
        let od4A =
            (MALTER
                od3A
                (31,24)
                ((ELEMENT (ASel(be t)) (3))
                     => (SUBARRAY (ASel(i_ad t)) (31,24))
                     |  (SUBARRAY (ASel(m_data_outQ t)) (31,24)))) in
        let od1B =
           (MALTER
               ARBN
               (7,0)
               ((ELEMENT (BSel(be t)) (0))
                    => (SUBARRAY (BSel(i_ad t)) (7,0))
                    |  (SUBARRAY (BSel(m_data_outQ t)) (7,0)))) in
        let od2B =
           (MALTER
               od1B
               (15,8)
               ((ELEMENT (BSel(be t)) (1))
                    => (SUBARRAY (BSel(i_ad t)) (15,8))
                    |  (SUBARRAY (BSel(m_data_outQ t)) (15,8)))) in
        let od3B =
           (MALTER
               od2B
               (23,16)
               ((ELEMENT (BSel(be t)) (2))
                    => (SUBARRAY (BSel(i_ad t)) (23,16))
                    |  (SUBARRAY (BSel(m_data_outQ t)) (23,16)))) in
        let od4B =
            (MALTER
                od3B
                (31,24)
                ((ELEMENT (BSel(be t)) (3))
                     => (SUBARRAY (BSel(i_ad t)) (31,24))
                     |  (SUBARRAY (BSel(m_data_outQ t)) (31,24)))) in
       (mb_data_out t = (od4A, od4B))"
   );;

%-----------------------------------------------------------------------------
Data encoding logic.
-----------------------------------------------------------------------------%

let Enc_Out_Logic_GATE = new_definition
   ('Enc_Out_Logic_GATE',
    "! (rep :^REP_ty)
       (mb_data_out mb_edata_out :time->wordn#wordn) .
     Enc_Out_Logic_GATE rep mb_data_out mb_edata_out =
        ! t:time .
        mb_edata_out t =
           ((Ham_Enc rep (ASel(mb_data_out t))),
            (Ham_Enc rep (BSel(mb_data_out t))))"
   );;

%-----------------------------------------------------------------------------
Input logic for M_parity latch.
-----------------------------------------------------------------------------%

let Memparity_In_Logic_GATE = new_definition
   ('Memparity_In_Logic_GATE',
    "! (srdy mem_enable detect_outQ rst reset_parity :time->bool#bool)
       (memparity_inS memparity_inR memparity_inE :time->bool#bool) .
     Memparity_In_Logic_GATE srdy mem_enable detect_outQ rst reset_parity
                             memparity_inS memparity_inR memparity_inE =
        ! t:time .
        (memparity_inS t =
           ((ASel(srdy t) /\ ASel(mem_enable t) /\ ASel (detect_outQ t)),
            (BSel(srdy t) /\ BSel(mem_enable t) /\ BSel (detect_outQ t)))) /\
        (memparity_inR t =
           ((ASel(reset_parity t) \/ ASel(rst t)),
            (BSel(reset_parity t) \/ BSel(rst t)))) /\
```

```
        (memparity_inE t =
           ((ASel(memparity_inS t) \/ ASel(memparity_inR t)),
            (BSel(memparity_inS t) \/ BSel(memparity_inR t))))"
   );;

%-----------------------------------------------------------------------------
M-Port controller state machine.
-----------------------------------------------------------------------------%

let FSM_GATE = new_definition
   ('FSM_GATE',
    "! (male_in_ rd_in bw_in ww_in last_in_ :time->bool#bool)
       (mrdy_in_ zero_cnt_in rst_in :time->bool#bool)
       (state :time->mfsm_ty)
       (male_ rd bw ww last_ mrdy_ zero_cnt rst :time->bool)
       (address_out read_out write_out :time->bool#bool)
       (byte_write_out mem_enable_out :time->bool#bool) .
    FSM_GATE male_in_ rd_in bw_in ww_in last_in_ mrdy_in_ zero_cnt_in rst_in
             state male_ rd bw ww last_ mrdy_ zero_cnt rst
             address_out read_out write_out byte_write_out mem_enable_out =
       ! t:time.
       (state (t+1) =
          (rst t) => MI |
          (state t = MI) => ((~male_ t) => MA | MI) |
          (state t = MA) =>
             (((~mrdy_ t) /\ ww t) => MW |
              ((~mrdy_ t) /\ (rd t \/ bw t)) => MR | MA) |
          (state t = MR) =>
             ((bw t /\ zero_cnt t) => MBW |
              (last_ t /\ rd t /\ zero_cnt t) => MA |
              ((~last_ t) /\ rd t /\ zero_cnt t) => MRR | MR) |
          (state t = MRR) => MI |
          (state t = MW) =>
             ((zero_cnt t /\ ~last_ t) => MI |
              (zero_cnt t /\ last_ t) => MA | MW) | MW) /\
       (male_ (t+1) = BSel(male_in_ t)) /\
       (rd (t+1) = BSel(rd_in t)) /\
       (bw (t+1) = BSel(bw_in t)) /\
       (ww (t+1) = BSel(ww_in t)) /\
       (last_ (t+1) = BSel(last_in_ t)) /\
       (mrdy_ (t+1) = BSel(mrdy_in_ t)) /\
       (zero_cnt (t+1) = BSel(zero_cnt_in t)) /\
       (rst (t+1) = BSel(rst_in t)) /\

       (address_out t = ((state (t+1) = MA), (state (t+1) = MA))) /\
       (read_out t = ((state (t+1) = MR), (state (t+1) = MR))) /\
       (write_out t = ((state (t+1) = MW), (state (t+1) = MW))) /\
       (byte_write_out t = ((state (t+1) = MBW), (state (t+1) = MBW))) /\
       (mem_enable_out t = ((~(state (t+1) = MI)), (~(state (t+1) = MI))))"
   );;

%-----------------------------------------------------------------------------
M-Port Block.
-----------------------------------------------------------------------------%

let MBlock_GATE = new_definition
   ('MBlock_GATE',
    "! (rep :^REP_ty) (s :time->mc_state) (e :time->mc_env) (p :time->mc_out) .
     MBlock_GATE rep s e p =
  ? (male mem_enable wr rd_mem wr_mem rdy_outQ srdy ww bw :time->bool#bool)
         (address read write byte_write zero_cnt rdy :time->bool#bool)
         (count_inDN count_inLD wwdel_inD wwdel_outQ edac_le :time->bool#bool)
         (mb_wr_en_ rdy_outQ srdy_ edac_en detect_inE :time->bool#bool)
         (memparity_inS memparity_inR memparity_inE :time->bool#bool)
         (m_detect_outQ :time->bool#bool)
         (be data_out detect_out mb_data_out mb_edata_out :time->wordn#wordn)
         (m_data_outQ :time->wordn#wordn) .
       (NOT_GATE (sig I_male_E e) male) /\
       (SE_Logic_GATE (sig I_ad_inE e) male mem_enable (sig M_seS s)
                      (sig MB_cs_eeprom_O p) (sig MB_cs_sram_O p)) /\
       (WR_Logic_GATE (sig I_ad_inE e) male mem_enable (sig M_wrS s) wr
                      rd_mem wr_mem) /\
```

```
        (Addr_Ctr_GATE (sig I_ad_inE e) male rdy_outQ (sig M_addrS s)
                       (sig MB_addrO p)) /\
        (BE_Logic_GATE (sig I_be_E e) male srdy wr_mem (sig M_beS s) be ww bw) /\
        (Rdy_Logic_GATE write read zero_cnt wr_mem rdy) /\
        (Ctr_Logic_GATE (sig MB_cs_eeprom_O p) count_inDN count_inLD
                        (sig M_countS s) zero_cnt) /\
        (OR2_GATE write read count_inDN) /\
        (OR2_GATE address byte_write count_inLD) /\
        (AND2_GATE ww address wwdel_inD) /\
        (DLatB_GATE wwdel_inD (sig M_wwdelS s) wwdel_outQ) /\
        (Enable_Logic_GATE (sig MB_cs_eeprom_O p) rd_mem address read write
                           byte_write wwdel_outQ (sig Disable_eepromE e)
                           (sig Disable_writesE e) (sig MB_oe_O p) edac_le
                           (sig MB_we_O p) mb_wr_en_) /\
        (DFFA_GATE rdy (sig M_rdyS s) rdy_outQ) /\
        (Srdy_Logic_GATE wr rdy rdy_outQ srdy_) /\
        (TRIBUF_GATE srdy_ mem_enable (sig I_srdy_O p)) /\
        (NOT_GATE srdy_ srdy) /\
        (NOT_GATE (sig Edac_en_E e) edac_en) /\
        (EDAC_Decode_Logic_GATE rep (sig MB_data_inE e) edac_en data_out
                                detect_out) /\
        (Read_Latches_GATE rep data_out edac_en edac_le detect_out detect_inE
                           (sig M_rd_dataS s) (sig M_detectS s) m_data_outQ
                           m_detect_outQ) /\
        (TRIBUFn_GATE m_data_outQ rd_mem (sig I_ad_outO p)) /\
        (Detect_Enable_Logic_GATE edac_en rd_mem detect_inE) /\
        (Mux_Out_Logic_GATE m_data_outQ (sig I_ad_inE e) be mb_data_out) /\
        (Enc_Out_Logic_GATE rep mb_data_out mb_edata_out) /\
        (TRINBUFn_GATE mb_edata_out mb_wr_en_ (sig MB_data_outO p)) /\
        (Memparity_In_Logic_GATE srdy mem_enable m_detect_outQ (sig RstE e)
                                 (sig Reset_parityE e) memparity_inS
                                 memparity_inR memparity_inE) /\
        (DSRELatB_GATE GND memparity_inS memparity_inR memparity_inE
                       (sig M_parityS s) (sig MB_parityO p)) /\
        (FSM_GATE (sig I_male_E e) rd_mem bw ww (sig I_last_E e) (sig I_mrdy_E e)
                  zero_cnt (sig RstE e) (sig M_fsm_stateS s) (sig M_fsm_male_S s)
                  (sig M_fsm_rdS s) (sig M_fsm_bwS s) (sig M_fsm_wwS s)
                  (sig M_fsm_last_S s) (sig M_fsm_mrdy_S s)
                  (sig M_fsm_zero_cntS s) (sig M_fsm_rstS s) address read write
                  byte_write mem_enable)"
   );;

let MBlock_EXP = save_thm
   ('MBlock_EXP',
    (BETA_RULE
    (REWRITE_RULE [SE_Logic_GATE;WR_Logic_GATE;Addr_Ctr_GATE;BE_Logic_GATE;
                   Rdy_Logic_GATE;Ctr_Logic_GATE;Enable_Logic_GATE;
                   Srdy_Logic_GATE;EDAC_Decode_Logic_GATE;Read_Latches_GATE;
                   Detect_Enable_Logic_GATE;(EXPAND_LET_RULE Mux_Out_Logic_GATE);
                   Enc_Out_Logic_GATE;Memparity_In_Logic_GATE;FSM_GATE;NOT_GATE;
                   OR2_GATE;AND2_GATE;TRIBUF_GATE;TRIBUFn_GATE;TRINBUFn_GATE;
                   DLatB_GATE; DSRELatB_GATE;DFFA_GATE;ASel;BSel;GND;sig]
    (SPEC_ALL MBlock_GATE)))
   );;


close_theory();;


%-----------------------------------------------------------------------------

   File:        mclock_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        1 March 1993

   This file contains the ml source for the clock-level specification of the PIU
   M-Port, an ASIC developed by the Embedded Processing Laboratory, Boeing High
   Technology Center.  The bulk of this code was translated from an M-language
   simulation program using a translator written by P.J. Windley at the
   University of Idaho.
```

```
------------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/ftep/piu/hol/mport/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

system 'rm mclock_def.th';;

new_theory 'mclock_def';;

loadf 'abs_theory';;
loadf 'aux_defs';;

map new_parent ['array_def';'wordn_def';'less_eq'];;
map load_parent ['piuaux_def';'maux_def'];;

new_type_abbrev ('timeC', ":num");;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%----------------------------------------------------------------------------
Next-state definition for M-Port instruction.
------------------------------------------------------------------------------%

let MC_NSF = new_definition
    ('MC_NSF',
     "! (rep :^REP_ty) (s :mc_state) (e :mc_env) .
      MC_NSF rep s e =
      let M_fsm_state = M_fsm_stateS s and
          M_fsm_male_ = M_fsm_male_S s and
          M_fsm_rd = M_fsm_rdS s and
          M_fsm_bw = M_fsm_bwS s and
          M_fsm_ww = M_fsm_wwS s and
          M_fsm_last_ = M_fsm_last_S s and
          M_fsm_mrdy_ = M_fsm_mrdy_S s and
          M_fsm_zero_cnt = M_fsm_zero_cntS s and
          M_fsm_rst = M_fsm_rstS s and
          M_se = M_seS s and
          M_wr = M_wrS s and
          M_addr = M_addrS s and
          M_be = M_beS s and
          M_count = M_countS s and
          M_rdy = M_rdyS s and
          M_wwdel = M_wwdelS s and
          M_parity = M_parityS s and
          M_rd_data = M_rd_dataS s and
          M_detect = M_detectS s in
      let Rst = RstE e and
          Disable_eeprom = Disable_eepromE e and
          Disable_writes = Disable_writesE e and
          I_ad_in = I_ad_inE e and
          I_male_ = I_male_E e and
          I_last_ = I_last_E e and
          I_be_ = I_be_E e and
          I_mrdy_ = I_mrdy_E e and
          MB_data_in = MB_data_inE e and
          Edac_en_ = Edac_en_E e and
          Reset_parity = Reset_parityE e in
      let new_M_fsm_state =
         ((M_fsm_rst) => MI |
          (M_fsm_state = MI) => ((~M_fsm_male_) => MA | MI) |
          (M_fsm_state = MA) =>
             (((~M_fsm_mrdy_) /\ M_fsm_ww) => MW |
              ((~M_fsm_mrdy_) /\ (M_fsm_rd \/ M_fsm_bw)) => MR | MA) |
          (M_fsm_state = MR) =>
             ((M_fsm_bw /\ M_fsm_zero_cnt) => MBW |
              (M_fsm_last_ /\ M_fsm_rd /\ M_fsm_zero_cnt) => MA |
              ((~M_fsm_last_) /\ M_fsm_rd /\ M_fsm_zero_cnt) => MRR | MR) |
```

```
     (M_fsm_state = MRR) => MI |
     (M_fsm_state = MW) =>
        ((M_fsm_zero_cnt /\ ~M_fsm_last_) => MI |
         (M_fsm_zero_cnt /\ M_fsm_last_) => MA | MW) | MW) in
 let address = (new_M_fsm_state = MA) in
 let read = (new_M_fsm_state = MR) in
 let write = (new_M_fsm_state = MW) in
 let byte_write = (new_M_fsm_state = MBW) in
 let mem_enable = (~(new_M_fsm_state = MI)) in
 let zero_cnt = (M_count = ((write \/ read ) => (WORDN 1 1) | (WORDN 1 0)))
 in
 let new_M_se =
    ((~BSel(I_male_)) => (ELEMENT (BSel(I_ad_in)) (23)) | M_se) in
 let new_M_wr =
    ((~BSel(I_male_)) => (ELEMENT (BSel(I_ad_in)) (27)) | M_wr) in
 let new_M_addr =
    ((~BSel(I_male_)) => (SUBARRAY (BSel(I_ad_in)) (18,0)) |
     (M_rdy) => (INCN 18 M_addr) | M_addr) in
 let rd_mem = ((mem_enable /\ ~M_wr),
               (mem_enable /\ ~new_M_wr)) in
 let wr_mem = ((mem_enable /\ M_wr),
               (mem_enable /\ new_M_wr)) in
 let rdy = (((write /\ zero_cnt) \/
            (read /\ zero_cnt /\ ~ASel(wr_mem))),
           ((write /\ zero_cnt) \/
            (read /\ zero_cnt /\ ~BSel(wr_mem)))) in
 let srdy_ =
    ((~((M_rdy /\ ~M_wr) \/ (ASel(rdy) /\ M_wr))),
     (~((M_rdy /\ ~new_M_wr) \/ (BSel(rdy) /\ new_M_wr)))) in
 let new_M_be =
    ((~BSel(I_male_) \/ ~BSel(srdy_)) => (NOTN 3 (BSel(I_be_))) | M_be) in
 let new_M_count =
    ((address \/ byte_write) =>
        ((~mem_enable \/ new_M_se) => (WORDN 1 1) | (WORDN 1 2)) |
     (write \/ read) => (DECN 1 M_count) | M_count) in
 let new_M_rdy = (BSel(rdy)) in
 let bw = ((ASel(wr_mem) /\ ~(M_be = (WORDN 3 15))),
           (BSel(wr_mem) /\ ~(new_M_be = (WORDN 3 15)))) in
 let ww = ((ASel(wr_mem) /\ (M_be = (WORDN 3 15))),
           (BSel(wr_mem) /\ (new_M_be = (WORDN 3 15)))) in
 let new_M_wwdel = (BSel(ww) /\ address) in
 let new_M_rd_data =
    (read => ((~BSel(Edac_en_)) => (Ham_Dec rep (BSel(MB_data_in)))
                                 |  BSel(MB_data_in))
          |  M_rd_data) in
 let new_M_detect =
    (((~BSel(Edac_en_) /\ BSel(rd_mem)) \/ ~BSel(rd_mem))
        => ((~BSel(Edac_en_)) => (Ham_Det1 rep (BSel(MB_data_in)))
                              |  (WORDN 3 0))
        |  M_detect) in
 let parityS =
    (~BSel(srdy_) /\ mem_enable
                  /\ Ham_Det2 rep (new_M_detect, ~BSel(Edac_en_))) in
 let parityR = (BSel(Reset_parity) \/ BSel(Rst)) in
 let new_M_parity =
    ((parityS \/ parityR)
        => ((parityS /\ ~parityR) => T |
    (~parityS /\ parityR) => F |
    (~parityS /\ ~parityR) => F | ARB)
        | M_parity) in
 let new_M_fsm_male_ = (BSel(I_male_)) in
 let new_M_fsm_rd = (BSel(rd_mem)) in
 let new_M_fsm_bw = (BSel(bw)) in
 let new_M_fsm_ww = (BSel(ww)) in
 let new_M_fsm_last_ = (BSel(I_last_)) in
 let new_M_fsm_mrdy_ = (BSel(I_mrdy_)) in
 let new_M_fsm_zero_cnt = zero_cnt in
 let new_M_fsm_rst = (BSel(Rst)) in

(MCState new_M_fsm_state new_M_fsm_male_ new_M_fsm_rd new_M_fsm_bw
         new_M_fsm_ww new_M_fsm_last_ new_M_fsm_mrdy_ new_M_fsm_zero_cnt
         new_M_fsm_rst new_M_se new_M_wr new_M_addr new_M_be new_M_count
```

```
            new_M_rdy new_M_wwdel new_M_parity new_M_rd_data new_M_detect)"
   );;

let MC_NSF_REW = save_thm
   ('MC_NSF_REW',
    (REWRITE_RULE [ASel;BSel] MC_NSF)
   );;

%-----------------------------------------------------------------------------
Output definition for M-Port instruction.
-----------------------------------------------------------------------------%

let MC_OF = new_definition
    ('MC_OF',
     "! (rep :^REP_ty) (s :mc_state) (e :mc_env) .
      MC_OF rep s e =
      let M_fsm_state = M_fsm_stateS s and
          M_fsm_male_ = M_fsm_male_S s and
          M_fsm_rd = M_fsm_rdS s and
          M_fsm_bw = M_fsm_bwS s and
          M_fsm_ww = M_fsm_wwS s and
          M_fsm_last_ = M_fsm_last_S s and
          M_fsm_mrdy_ = M_fsm_mrdy_S s and
          M_fsm_zero_cnt = M_fsm_zero_cntS s and
          M_fsm_rst = M_fsm_rstS s and
          M_se = M_seS s and
          M_wr = M_wrS s and
          M_addr = M_addrS s and
          M_be = M_beS s and
          M_count = M_countS s and
          M_rdy = M_rdyS s and
          M_wwdel = M_wwdelS s and
          M_parity = M_parityS s and
          M_rd_data = M_rd_dataS s and
          M_detect = M_detectS s in
      let Rst = RstE e and
          Disable_eeprom = Disable_eepromE e and
          Disable_writes = Disable_writesE e and
          I_ad_in = I_ad_inE e and
          I_male_ = I_male_E e and
          I_last_ = I_last_E e and
          I_be_ = I_be_E e and
          I_mrdy_ = I_mrdy_E e and
          MB_data_in = MB_data_inE e and
          Edac_en_ = Edac_en_E e and
          Reset_parity = Reset_parityE e in
      let new_M_fsm_state =
         ((M_fsm_rst) => MI |
          (M_fsm_state = MI) => ((~M_fsm_male_) => MA | MI) |
          (M_fsm_state = MA) =>
             (((~M_fsm_mrdy_) /\ M_fsm_ww) => MW |
              ((~M_fsm_mrdy_) /\ (M_fsm_rd \/ M_fsm_bw)) => MR | MA) |
          (M_fsm_state = MR) =>
             ((M_fsm_bw /\ M_fsm_zero_cnt) => MBW |
              (M_fsm_last_ /\ M_fsm_rd /\ M_fsm_zero_cnt) => MA |
              ((~M_fsm_last_) /\ M_fsm_rd /\ M_fsm_zero_cnt) => MRR | MR) |
          (M_fsm_state = MRR) => MI |
          (M_fsm_state = MW) =>
             ((M_fsm_zero_cnt /\ ~M_fsm_last_) => MI |
              (M_fsm_zero_cnt /\ M_fsm_last_) => MA | MW) | MW) in
      let address = (new_M_fsm_state = MA) in
      let read = (new_M_fsm_state = MR) in
      let write = (new_M_fsm_state = MW) in
      let byte_write = (new_M_fsm_state = MBW) in
      let mem_enable = (~(new_M_fsm_state = MI)) in
      let zero_cnt = (M_count = ((write \/ read ) => (WORDN 1 1) | (WORDN 1 0)))
      in
      let new_M_se =
         ((~BSel(I_male_)) => (ELEMENT (BSel(I_ad_in)) (23)) | M_se) in
      let new_M_wr =
         ((~BSel(I_male_)) => (ELEMENT (BSel(I_ad_in)) (27)) | M_wr) in
      let new_M_addr =
```

```
   ((~BSel(I_male_)) => (SUBARRAY (BSel(I_ad_in)) (18,0)) |
    (M_rdy) => (INCN 18 M_addr) | M_addr) in
let rd_mem = ((mem_enable /\ ~M_wr),
              (mem_enable /\ ~new_M_wr)) in
let wr_mem = ((mem_enable /\ M_wr),
              (mem_enable /\ new_M_wr)) in
let rdy = (((write /\ zero_cnt) \/
            (read /\ zero_cnt /\ ~ASel(wr_mem))),
           ((write /\ zero_cnt) \/
            (read /\ zero_cnt /\ ~BSel(wr_mem)))) in
let srdy_ =
   ((~((M_rdy /\ ~M_wr) \/ (ASel(rdy) /\ M_wr))),
    (~((M_rdy /\ ~new_M_wr) \/ (BSel(rdy) /\ new_M_wr)))) in
let new_M_be =
   ((~BSel(I_male_) \/ ~BSel(srdy_)) => (NOTN 3 (BSel(I_be_))) | M_be) in
let new_M_count =
   ((address \/ byte_write) =>
       ((~mem_enable \/ new_M_se) => (WORDN 1 1) | (WORDN 1 2)) |
    (write \/ read) => (DECN 1 M_count) | M_count) in
let new_M_rdy = (BSel(rdy)) in
let bw = ((ASel(wr_mem) /\ ~(M_be = (WORDN 3 15))),
          (BSel(wr_mem) /\ ~(new_M_be = (WORDN 3 15)))) in
let ww = ((ASel(wr_mem) /\ (M_be = (WORDN 3 15))),
          (BSel(wr_mem) /\ (new_M_be = (WORDN 3 15)))) in
let new_M_wwdel = (BSel(ww) /\ address) in
let new_M_rd_data =
   (read => ((~BSel(Edac_en_)) => (Ham_Dec rep (BSel(MB_data_in)))
                                | BSel(MB_data_in))
         | M_rd_data) in
let new_M_detect =
   (((~BSel(Edac_en_) /\ BSel(rd_mem)) \/ ~BSel(rd_mem))
       => ((~BSel(Edac_en_)) => (Ham_Det1 rep (BSel(MB_data_in)))
                              | (WORDN 3 0))
       | M_detect) in
let parityS =
   (~BSel(srdy_) /\ mem_enable
                 /\ Ham_Det2 rep (new_M_detect, ~BSel(Edac_en_))) in
let parityR = (BSel(Reset_parity) \/ BSel(Rst)) in
let new_M_parity =
   ((parityS \/ parityR)
       => ((parityS /\ ~parityR) => T |
   (~parityS /\ parityR) => F |
   (~parityS /\ ~parityR) => F | ARB)
       | M_parity) in
let new_M_fsm_male_ = (BSel(I_male_)) in
let new_M_fsm_rd = (BSel(rd_mem)) in
let new_M_fsm_bw = (BSel(bw)) in
let new_M_fsm_ww = (BSel(ww)) in
let new_M_fsm_last_ = (BSel(I_last_)) in
let new_M_fsm_mrdy_ = (BSel(I_mrdy_)) in
let new_M_fsm_zero_cnt = zero_cnt in
let new_M_fsm_rst = (BSel(Rst)) in
let I_ad_out = (((ASel(rd_mem)) => BUSN M_rd_data | Offn),
                ((BSel(rd_mem)) => BUSN M_rd_data | Offn)) in
let I_srdy_ = (((mem_enable) => WIRE (ASel(srdy_)) | Z),
               ((mem_enable) => WIRE (BSel(srdy_)) | Z)) in
let MB_addr =
   (((M_rdy) => (INCN 18 M_addr) | M_addr),
    ((M_rdy) => (INCN 18 M_addr) | M_addr)) in
let od1A =
   (MALTER
      ARBN
      (7,0)
      ((ELEMENT M_be (0))
          => (SUBARRAY (ASel(I_ad_in)) (7,0))
          | (SUBARRAY M_rd_data (7,0)))) in
let od2A =
   (MALTER
      od1A
      (15,8)
      ((ELEMENT M_be (1))
          => (SUBARRAY (ASel(I_ad_in)) (15,8))
```

```
            |  (SUBARRAY M_rd_data (15,8)))) in
    let od3A =
       (MALTER
          od2A
          (23,16)
          ((ELEMENT M_be (2))
              => (SUBARRAY (ASel(I_ad_in)) (23,16))
              |  (SUBARRAY M_rd_data (23,16)))) in
    let od4A =
       (MALTER
          od3A
          (31,24)
          ((ELEMENT M_be (3))
              => (SUBARRAY (ASel(I_ad_in)) (31,24))
              |  (SUBARRAY M_rd_data (31,24)))) in
    let od1B =
       (MALTER
          ARBN
          (7,0)
          ((ELEMENT M_be (0))
              => (SUBARRAY (BSel(I_ad_in)) (7,0))
              |  (SUBARRAY M_rd_data (7,0)))) in
    let od2B =
       (MALTER
          od1B
          (15,8)
          ((ELEMENT M_be (1))
              => (SUBARRAY (BSel(I_ad_in)) (15,8))
              |  (SUBARRAY M_rd_data (15,8)))) in
    let od3B =
       (MALTER
          od2B
          (23,16)
          ((ELEMENT M_be (2))
              => (SUBARRAY (BSel(I_ad_in)) (23,16))
              |  (SUBARRAY M_rd_data (23,16)))) in
    let od4B =
       (MALTER
          od3B
          (31,24)
          ((ELEMENT M_be (3))
              => (SUBARRAY (BSel(I_ad_in)) (31,24))
              |  (SUBARRAY M_rd_data (31,24)))) in
    let MB_data_out = (((write) => BUSN (Ham_Enc rep od4A) | Offn),
                       ((write) => BUSN (Ham_Enc rep od4B) | Offn)) in
    let MB_cs_eeprom_ = ((~mem_enable \/ M_se), (~mem_enable \/ new_M_se)) in
    let MB_cs_sram_ = ((~mem_enable \/ ~M_se), (~mem_enable \/ ~new_M_se)) in
    let MB_we_ =
       ((~((ASel(MB_cs_eeprom_) \/ ~ASel(Disable_eeprom)) /\
           ~ASel(Disable_writes) /\
           (byte_write \/ write \/ M_wwdel))),
        (~((BSel(MB_cs_eeprom_) \/ ~BSel(Disable_eeprom)) /\
           ~BSel(Disable_writes) /\
           (byte_write \/ write \/ new_M_wwdel)))) in
    let MB_oe_ = ((~((ASel(rd_mem) /\ address) \/ read)),
                  (~((BSel(rd_mem) /\ address) \/ read))) in
    let MB_parity = (M_parity, new_M_parity) in

    (MCOut I_ad_out I_srdy_ MB_addr MB_data_out MB_cs_eeprom_ MB_cs_sram_
           MB_we_ MB_oe_ MB_parity)"
  );;

let MC_OF_REW = save_thm
   ('MC_OF_REW',
    (REWRITE_RULE [ASel;BSel] MC_OF)
   );;

let MC_Exec = new_definition
   ('MC_Exec',
    "! (mci :MCI) (s :timeC->mc_state) (e :timeC->mc_env) (p :timeC->mc_out)
       (t :timeC) .
     MC_Exec mci s e p t = T"
```

```
   );;

let MC_PreC = new_definition
   ('MC_PreC',
    "! (mci :MCI) (s :timeC->mc_state) (e :timeC->mc_env) (p :timeC->mc_out)
       (t :timeC) .
     MC_PreC mci s e p t = T"
   );;

let MC_PostC = new_definition
   ('MC_PostC',
    "! (rep :^REP_ty) (mci :MCI) (s :timeC->mc_state) (e :timeC->mc_env)
       (p :timeC->mc_out) (t :timeC) .
     MC_PostC rep mci s e p t =
        (s (t+1) = MC_NSF rep (s t) (e t)) /\
        (p t = MC_OF rep (s t) (e t))"
   );;

let MC_Correct = new_definition
   ('MC_Correct',
    "! (rep :^REP_ty) (mci :MCI) (s :timeC->mc_state) (e :timeC->mc_env)
       (p :timeC->mc_out) (t :timeC) .
     MC_Correct rep mci s e p t =
        MC_Exec mci s e p t /\
        MC_PreC mci s e p t
        ==>
        MC_PostC rep mci s e p t"
   );;

let MCSet_Correct = new_definition
   ('MCSet_Correct',
    "! (rep :^REP_ty) (s :timeC->mc_state) (e :timeC->mc_env) (p :timeC->mc_out).
     MCSet_Correct rep s e p = !(mci:MCI)(t:timeC). MC_Correct rep mci s e p t"
   );;

let Next_State_THM = TAC_PROOF
   (([], "MCSet_Correct rep s e p ==> (s (t+1) = MC_NSF rep (s t) (e t))"),
    REWRITE_TAC [MCSet_Correct;MC_Correct;MC_Exec;MC_PreC;MC_PostC]
    THEN REWRITE_TAC
       [LEFT_IMP_FORALL_CONV
          "(!t.
            (s(t + 1) = MC_NSF rep (s t)(e t)) /\
            (p t = MC_OF rep (s t)(e t))) ==>
            (s (t+1) = MC_NSF rep (s t)(e t))"]
    THEN EXISTS_TAC "t:timeC"
    THEN REWRITE_TAC [ADD_CLAUSES]
    THEN STRIP_TAC
   );;

let M_fsm_state_ISO = save_thm
   ('M_fsm_state_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_stateS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_stateS (s ((t:timeC) + 1))")))))
   );;

let M_fsm_male_ISO = save_thm
   ('M_fsm_male_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_male_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_male_S (s ((t:timeC) + 1))")))))
   );;

let M_fsm_rd_ISO = save_thm
   ('M_fsm_rd_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_rdS]
```

```
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_rdS (s ((t:timeC) + 1))"))))
   );;

let M_fsm_bw_ISO = save_thm
   ('M_fsm_bw_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_bwS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_bwS (s ((t:timeC) + 1))"))))
   );;

let M_fsm_ww_ISO = save_thm
   ('M_fsm_ww_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_wwS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_wwS (s ((t:timeC) + 1))"))))
   );;

let M_fsm_last_ISO = save_thm
   ('M_fsm_last_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_last_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_last_S (s ((t:timeC) + 1))"))))
   );;

let M_fsm_mrdy_ISO = save_thm
   ('M_fsm_mrdy_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_mrdy_S]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_mrdy_S (s ((t:timeC) + 1))"))))
   );;

let M_fsm_zero_cnt_ISO = save_thm
   ('M_fsm_zero_cnt_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_zero_cntS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_zero_cntS (s ((t:timeC) + 1))"))))
   );;

let M_fsm_rst_ISO = save_thm
   ('M_fsm_rst_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_fsm_rstS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_fsm_rstS (s ((t:timeC) + 1))"))))
   );;

let M_se_ISO = save_thm
   ('M_se_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_seS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_seS (s ((t:timeC) + 1))"))))
   );;

let M_wr_ISO = save_thm
   ('M_wr_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_wrS]
```

```
      (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                 (REFL "M_wrS (s ((t:timeC) + 1))"))))
   );;

let M_addr_ISO = save_thm
   ('M_addr_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_addrS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_addrS (s ((t:timeC) + 1))"))))
   );;

let M_be_ISO = save_thm
   ('M_be_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_beS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_beS (s ((t:timeC) + 1))"))))
   );;

let M_count_ISO = save_thm
   ('M_count_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_countS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_countS (s ((t:timeC) + 1))"))))
   );;

let M_rdy_ISO = save_thm
   ('M_rdy_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_rdyS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_rdyS (s ((t:timeC) + 1))"))))
   );;

let M_wwdel_ISO = save_thm
   ('M_wwdel_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_wwdelS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_wwdelS (s ((t:timeC) + 1))"))))
   );;

let M_parity_ISO = save_thm
   ('M_parity_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_parityS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_parityS (s ((t:timeC) + 1))"))))
   );;

let M_rd_data_ISO = save_thm
   ('M_rd_data_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_rd_dataS]
       (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                  (REFL "M_rd_dataS (s ((t:timeC) + 1))"))))
   );;

let M_detect_ISO = save_thm
   ('M_detect_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_NSF_REW;M_detectS]
```

```
      (SUBS_OCCS [([2],UNDISCH (Next_State_THM))]
                 (REFL "M_detectS (s ((t:timeC) + 1))"))))
  );;

let Output_THM = TAC_PROOF
  (([], "MCSet_Correct rep s e p ==> (p t = MC_OF rep (s t) (e t))"),
   REWRITE_TAC [MCSet_Correct;MC_Correct;MC_Exec;MC_PreC;MC_PostC]
   THEN REWRITE_TAC
      [LEFT_IMP_FORALL_CONV
         "(!t.
           (s(t + 1) = MC_NSF rep (s t)(e t)) /\
           (p t = MC_OF rep (s t)(e t))) ==>
           (p t = MC_OF rep (s t)(e t))"]
   THEN EXISTS_TAC "t:timeC"
   THEN REWRITE_TAC [ADD_CLAUSES]
   THEN STRIP_TAC
  );;

let I_ad_out_ISO = save_thm
  ('I_ad_out_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [EXPAND_LET_RULE MC_OF_REW;I_ad_outO]
      (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                 (REFL "I_ad_outO (p (t:timeC))")))))
  );;

let I_srdy_ISO = save_thm
  ('I_srdy_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [EXPAND_LET_RULE MC_OF_REW;I_srdy_O]
      (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                 (REFL "I_srdy_O (p (t:timeC))")))))
  );;

let MB_addr_ISO = save_thm
  ('MB_addr_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [EXPAND_LET_RULE MC_OF_REW;MB_addrO]
      (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                 (REFL "MB_addrO (p (t:timeC))")))))
  );;

let MB_data_out_ISO = save_thm
  ('MB_data_out_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [EXPAND_LET_RULE MC_OF_REW;MB_data_outO]
      (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                 (REFL "MB_data_outO (p (t:timeC))")))))
  );;

let MB_cs_eeprom_ISO = save_thm
  ('MB_cs_eeprom_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [EXPAND_LET_RULE MC_OF_REW;MB_cs_eeprom_O]
      (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                 (REFL "MB_cs_eeprom_O (p (t:timeC))")))))
  );;

let MB_cs_sram_ISO = save_thm
  ('MB_cs_sram_ISO',
   (DISCH_ALL
    (REWRITE_RULE
      [EXPAND_LET_RULE MC_OF_REW;MB_cs_sram_O]
      (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                 (REFL "MB_cs_sram_O (p (t:timeC))")))))
  );;
```

```
let MB_we_ISO = save_thm
   ('MB_we_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_OF_REW;MB_we_O]
       (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                  (REFL "MB_we_O (p (t:timeC))"))))
   );;

let MB_oe_ISO = save_thm
   ('MB_oe_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_OF_REW;MB_oe_O]
       (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                  (REFL "MB_oe_O (p (t:timeC))"))))
   );;

let MB_parity_ISO = save_thm
   ('MB_parity_ISO',
    (DISCH_ALL
     (REWRITE_RULE
       [EXPAND_LET_RULE MC_OF_REW;MB_parityO]
       (SUBS_OCCS [([2],UNDISCH (Output_THM))]
                  (REFL "MB_parityO (p (t:timeC))"))))
   );;

close_theory();;
```

3.4 R-Port Definitions

This section contains the theories *raux_def*, *rblock_def*, and *rclock_def*, defining the R-Port design.

```
%-----------------------------------------------------------------------------

   File:         raux_def.ml

   Author:       (c) D.A. Fura 1992-93

   Date:         4 March 1993

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing', true);;

system 'rm raux_def.th';;

new_theory 'raux_def';;

map new_parent ['busn_def';'ineq'];;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('wordn', ":num->bool");;
new_type_abbrev ('busn', ":num->wire");;

%-----------------------------------------------------------------------------
   Abstract data type for the R-Port FSM states.
-----------------------------------------------------------------------------%

let rfsm_ty_Axiom =
   define_type 'rfsm_ty_Axiom'
               'rfsm_ty = RI | RA | RD';;
```

```
%-----------------------------------------------------------------------------
   Abstract data type for the R-Port instruction.
-----------------------------------------------------------------------------%
let RCI =
   define_type 'RCI'
               'RCI = RC_X';;

%-----------------------------------------------------------------------------
   Abstract data type for the state.
-----------------------------------------------------------------------------%

let r_state =
   define_type 'r_state'
               'r_state = RState wordn wordn wordn wordn wordn rfsm_ty bool bool
                                bool bool bool bool bool bool bool bool bool
                                bool bool wordn wordn bool bool bool wordn
                                wordn bool wordn bool bool bool wordn wordn
                                bool wordn bool bool bool wordn wordn bool
                                wordn bool bool bool wordn wordn bool bool
                                wordn wordn wordn bool wordn bool wordn bool
                                wordn bool';;

let R_ctr0S = new_recursive_definition
    false
    r_state
    'R_ctr0S'
    "R_ctr0S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                     R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                     R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                     R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                     R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                     R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                     R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                     R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                     R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                     R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                     R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                     R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                     R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0";;

let R_ctr1S = new_recursive_definition
    false
    r_state
    'R_ctr1S'
    "R_ctr1S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                     R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                     R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                     R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                     R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                     R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                     R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                     R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                     R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                     R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                     R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                     R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                     R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1";;

let R_ctr2S = new_recursive_definition
    false
    r_state
    'R_ctr2S'
    "R_ctr2S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                     R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                     R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                     R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                     R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                     R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                     R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                     R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
```

```
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2";;

let R_ctr3S = new_recursive_definition
    false
    r_state
    'R_ctr3S'
    "R_ctr3S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3";;

let R_busA_latchS = new_recursive_definition
    false
    r_state
    'R_busA_latchS'
    "R_busA_latchS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_busA_latch";;

let R_fsm_stateS = new_recursive_definition
    false
    r_state
    'R_fsm_stateS'
    "R_fsm_stateS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_fsm_state";;

let R_fsm_ale_S = new_recursive_definition
    false
    r_state
    'R_fsm_ale_S'
    "R_fsm_ale_S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
```

```
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_fsm_ale_";;

let R_fsm_mrdy_S = new_recursive_definition
    false
    r_state
    'R_fsm_mrdy_S'
    "R_fsm_mrdy_S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_fsm_mrdy_";;

let R_fsm_last_S = new_recursive_definition
    false
    r_state
    'R_fsm_last_S'
    "R_fsm_last_S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_fsm_last_";;

let R_fsm_rstS = new_recursive_definition
    false
    r_state
    'R_fsm_rstS'
    "R_fsm_rstS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_fsm_rst";;

let R_int0_disS = new_recursive_definition
    false
    r_state
    'R_int0_disS'
```

```
    "R_int0_disS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_int0_dis";;

let R_int3_disS = new_recursive_definition
    false
    r_state
    'R_int3_disS'
    "R_int3_disS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_int3_dis";;

let R_c01_cout_delS = new_recursive_definition
    false
    r_state
    'R_c01_cout_delS'
    "R_c01_cout_delS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_c01_cout_del";;

let R_int1_enS = new_recursive_definition
    false
    r_state
    'R_int1_enS'
    "R_int1_enS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_int1_en";;
```

```
let R_c23_cout_delS = new_recursive_definition
    false
    r_state
    'R_c23_cout_delS'
    "R_c23_cout_delS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                      R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                      R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                      R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                      R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                      R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                      R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                      R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                      R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                      R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                      R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                      R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                      R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_c23_cout_del";;

let R_int2_enS = new_recursive_definition
    false
    r_state
    'R_int2_enS'
    "R_int2_enS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                 R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                 R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                 R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                 R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                 R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                 R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                 R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                 R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                 R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                 R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                 R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                 R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_int2_en";;

let R_wrS = new_recursive_definition
    false
    r_state
    'R_wrS'
    "R_wrS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                 R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                 R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                 R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                 R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                 R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                 R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                 R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                 R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                 R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                 R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                 R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                 R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_wr";;

let R_cntlatch_delS = new_recursive_definition
    false
    r_state
    'R_cntlatch_delS'
    "R_cntlatch_delS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                      R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                      R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                      R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                      R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                      R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                      R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                      R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                      R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                      R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                      R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
```

```
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_cntlatch_del";;

let R_srdy_del_S = new_recursive_definition
    false
    r_state
    'R_srdy_del_S'
    "R_srdy_del_S (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_srdy_del_";;

let R_reg_selS = new_recursive_definition
    false
    r_state
    'R_reg_selS'
    "R_reg_selS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_reg_sel";;

let R_ctr0_inS = new_recursive_definition
    false
    r_state
    'R_ctr0_inS'
    "R_ctr0_inS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0_in";;

let R_ctr0_mux_selS = new_recursive_definition
    false
    r_state
    'R_ctr0_mux_selS'
    "R_ctr0_mux_selS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
```

```
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0_mux_sel";;

let R_ctr0_irdenS = new_recursive_definition
    false
    r_state
    'R_ctr0_irdenS'
    "R_ctr0_irdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0_irden";;

let R_ctr0_cryS = new_recursive_definition
    false
    r_state
    'R_ctr0_cryS'
    "R_ctr0_cryS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0_cry";;

let R_ctr0_newS = new_recursive_definition
    false
    r_state
    'R_ctr0_newS'
    "R_ctr0_newS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0_new";;

let R_ctr0_outS = new_recursive_definition
    false
    r_state
    'R_ctr0_outS'
    "R_ctr0_outS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
```

```
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0_out";;

let R_ctr0_ordenS = new_recursive_definition
    false
    r_state
    'R_ctr0_ordenS'
    "R_ctr0_ordenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr0_orden";;

let R_ctr1_inS = new_recursive_definition
    false
    r_state
    'R_ctr1_inS'
    "R_ctr1_inS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1_in";;

let R_ctr1_mux_selS = new_recursive_definition
    false
    r_state
    'R_ctr1_mux_selS'
    "R_ctr1_mux_selS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1_mux_sel";;

let R_ctr1_irdenS = new_recursive_definition
    false
    r_state
```

```
    'R_ctr1_irdenS'
    "R_ctr1_irdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                    R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                    R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                    R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                    R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                    R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                    R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                    R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                    R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                    R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                    R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                    R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                    R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1_irden";;

let R_ctr1_cryS = new_recursive_definition
    false
    r_state
    'R_ctr1_cryS'
    "R_ctr1_cryS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                  R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                  R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                  R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                  R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                  R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                  R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                  R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                  R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                  R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                  R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                  R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                  R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1_cry";;

let R_ctr1_newS = new_recursive_definition
    false
    r_state
    'R_ctr1_newS'
    "R_ctr1_newS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                  R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                  R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                  R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                  R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                  R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                  R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                  R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                  R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                  R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                  R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                  R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                  R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1_new";;

let R_ctr1_outS = new_recursive_definition
    false
    r_state
    'R_ctr1_outS'
    "R_ctr1_outS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                  R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                  R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                  R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                  R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                  R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                  R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                  R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                  R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                  R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                  R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                  R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                  R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1_out";;
```

```
let R_ctr1_ordenS = new_recursive_definition
    false
    r_state
    'R_ctr1_ordenS'
    "R_ctr1_ordenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr1_orden";;

let R_ctr2_inS = new_recursive_definition
    false
    r_state
    'R_ctr2_inS'
    "R_ctr2_inS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2_in";;

let R_ctr2_mux_selS = new_recursive_definition
    false
    r_state
    'R_ctr2_mux_selS'
    "R_ctr2_mux_selS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2_mux_sel";;

let R_ctr2_irdenS = new_recursive_definition
    false
    r_state
    'R_ctr2_irdenS'
    "R_ctr2_irdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
```

```
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2_irden";;

let R_ctr2_cryS = new_recursive_definition
    false
    r_state
    'R_ctr2_cryS'
    "R_ctr2_cryS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2_cry";;

let R_ctr2_newS = new_recursive_definition
    false
    r_state
    'R_ctr2_newS'
    "R_ctr2_newS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2_new";;

let R_ctr2_outS = new_recursive_definition
    false
    r_state
    'R_ctr2_outS'
    "R_ctr2_outS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2_out";;

let R_ctr2_ordenS = new_recursive_definition
    false
    r_state
    'R_ctr2_ordenS'
    "R_ctr2_ordenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
```

```
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr2_orden";;

let R_ctr3_inS = new_recursive_definition
    false
    r_state
    'R_ctr3_inS'
    "R_ctr3_inS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3_in";;

let R_ctr3_mux_selS = new_recursive_definition
    false
    r_state
    'R_ctr3_mux_selS'
    "R_ctr3_mux_selS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3_mux_sel";;

let R_ctr3_irdenS = new_recursive_definition
    false
    r_state
    'R_ctr3_irdenS'
    "R_ctr3_irdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3_irden";;

let R_ctr3_cryS = new_recursive_definition
    false
    r_state
    'R_ctr3_cryS'
    "R_ctr3_cryS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
```

```
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3_cry";;

let R_ctr3_newS = new_recursive_definition
    false
    r_state
    'R_ctr3_newS'
    "R_ctr3_newS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3_new";;

let R_ctr3_outS = new_recursive_definition
    false
    r_state
    'R_ctr3_outS'
    "R_ctr3_outS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3_out";;

let R_ctr3_ordenS = new_recursive_definition
    false
    r_state
    'R_ctr3_ordenS'
    "R_ctr3_ordenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ctr3_orden";;

let R_icr_loadS = new_recursive_definition
    false
```

```
    r_state
    'R_icr_loadS'
    "R_icr_loadS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_icr_load";;

let R_icr_oldS = new_recursive_definition
    false
    r_state
    'R_icr_oldS'
    "R_icr_oldS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_icr_old";;

let R_icr_maskS = new_recursive_definition
    false
    r_state
    'R_icr_maskS'
    "R_icr_maskS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_icr_mask";;

let R_icrS = new_recursive_definition
    false
    r_state
    'R_icrS'
    "R_icrS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
```

```
    = R_icr";;

let R_icr_rdenS = new_recursive_definition
    false
    r_state
    'R_icr_rdenS'
    "R_icr_rdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_icr_rden";;

let R_ccrS = new_recursive_definition
    false
    r_state
    'R_ccrS'
    "R_ccrS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ccr";;

let R_ccr_rdenS = new_recursive_definition
    false
    r_state
    'R_ccr_rdenS'
    "R_ccr_rdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                         R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                         R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                         R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                         R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_ccr_rden";;

let R_gcrS = new_recursive_definition
    false
    r_state
    'R_gcrS'
    "R_gcrS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                         R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                         R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                         R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                         R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                         R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                         R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                         R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                         R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
```

```
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_gcr";;

let R_gcr_rdenS = new_recursive_definition
    false
    r_state
    'R_gcr_rdenS'
    "R_gcr_rdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_gcr_rden";;

let R_srS = new_recursive_definition
    false
    r_state
    'R_srS'
    "R_srS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_sr";;

let R_sr_rdenS = new_recursive_definition
    false
    r_state
    'R_sr_rdenS'
    "R_sr_rdenS (RState R_ctr0 R_ctr1 R_ctr2 R_ctr3 R_busA_latch R_fsm_state
                        R_fsm_ale_ R_fsm_mrdy_ R_fsm_last_ R_fsm_rst R_int0_dis
                        R_int3_dis R_c01_cout_del R_int1_en R_c23_cout_del
                        R_int2_en R_wr R_cntlatch_del R_srdy_del_ R_reg_sel
                        R_ctr0_in R_ctr0_mux_sel R_ctr0_irden R_ctr0_cry
                        R_ctr0_new R_ctr0_out R_ctr0_orden R_ctr1_in
                        R_ctr1_mux_sel R_ctr1_irden R_ctr1_cry R_ctr1_new
                        R_ctr1_out R_ctr1_orden R_ctr2_in R_ctr2_mux_sel
                        R_ctr2_irden R_ctr2_cry R_ctr2_new R_ctr2_out
                        R_ctr2_orden R_ctr3_in R_ctr3_mux_sel R_ctr3_irden
                        R_ctr3_cry R_ctr3_new R_ctr3_out R_ctr3_orden
                        R_icr_load R_icr_old R_icr_mask R_icr R_icr_rden R_ccr
                        R_ccr_rden R_gcr R_gcr_rden R_sr R_sr_rden)
    = R_sr_rden";;

let State_CASES =
   prove_cases_thm (prove_induction_thm r_state);;

let State_Selectors_Work = prove_thm
   ('State_Selectors_Work',
    "!s:r_state.
     s = (RState (R_ctr0S s) (R_ctr1S s) (R_ctr2S s) (R_ctr3S s)
                 (R_busA_latchS s) (R_fsm_stateS s) (R_fsm_ale_S s)
                 (R_fsm_mrdy_S s) (R_fsm_last_S s) (R_fsm_rstS s)
```

```
                (R_int0_disS s) (R_int3_disS s) (R_c01_cout_delS s)
                (R_int1_enS s) (R_c23_cout_delS s) (R_int2_enS s) (R_wrS s)
                (R_cntlatch_delS s) (R_srdy_del_S s) (R_reg_selS s)
                (R_ctr0_inS s) (R_ctr0_mux_selS s) (R_ctr0_irdenS s)
                (R_ctr0_cryS s) (R_ctr0_newS s) (R_ctr0_outS s)
                (R_ctr0_ordenS s) (R_ctr1_inS s) (R_ctr1_mux_selS s)
                (R_ctr1_irdenS s) (R_ctr1_cryS s) (R_ctr1_newS s)
                (R_ctr1_outS s) (R_ctr1_ordenS s) (R_ctr2_inS s)
                (R_ctr2_mux_selS s) (R_ctr2_irdenS s) (R_ctr2_cryS s)
                (R_ctr2_newS s) (R_ctr2_outS s) (R_ctr2_ordenS s)
                (R_ctr3_inS s) (R_ctr3_mux_selS s) (R_ctr3_irdenS s)
                (R_ctr3_cryS s) (R_ctr3_newS s) (R_ctr3_outS s)
                (R_ctr3_ordenS s) (R_icr_loadS s) (R_icr_oldS s)
                (R_icr_maskS s) (R_icrS s) (R_icr_rdenS s) (R_ccrS s)
                (R_ccr_rdenS s) (R_gcrS s) (R_gcr_rdenS s) (R_srS s)
                (R_sr_rdenS s))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "s:r_state" State_CASES)
    THEN REWRITE_TAC [R_ctr0S; R_ctr1S; R_ctr2S; R_ctr3S;
                      R_busA_latchS; R_fsm_stateS; R_fsm_ale_S;
                      R_fsm_mrdy_S; R_fsm_last_S; R_fsm_rstS; R_int0_disS;
                      R_int3_disS; R_c01_cout_delS; R_int1_enS; R_c23_cout_delS;
                      R_int2_enS; R_wrS; R_cntlatch_delS; R_srdy_del_S;
                      R_reg_selS; R_ctr0_inS; R_ctr0_mux_selS; R_ctr0_irdenS;
                      R_ctr0_cryS; R_ctr0_newS; R_ctr0_outS; R_ctr0_ordenS;
                      R_ctr1_inS; R_ctr1_mux_selS; R_ctr1_irdenS; R_ctr1_cryS;
                      R_ctr1_newS; R_ctr1_outS; R_ctr1_ordenS; R_ctr2_inS;
                      R_ctr2_mux_selS; R_ctr2_irdenS; R_ctr2_cryS; R_ctr2_newS;
                      R_ctr2_outS; R_ctr2_ordenS; R_ctr3_inS; R_ctr3_mux_selS;
                      R_ctr3_irdenS; R_ctr3_cryS; R_ctr3_newS; R_ctr3_outS;
                      R_ctr3_ordenS; R_icr_loadS; R_icr_oldS; R_icr_maskS;
                      R_icrS; R_icr_rdenS; R_ccrS; R_ccr_rdenS; R_gcrS;
                      R_gcr_rdenS; R_srS; R_sr_rdenS]
  );;

%----------------------------------------------------------------------------
   Abstract data type for the environment.
----------------------------------------------------------------------------%

let r_env =
   define_type 'r_env'
               'r_env = REnv bool#bool wordn#wordn bool#bool bool#bool
                             wordn#wordn bool#bool bool#bool bool#bool
                             wordn#wordn wordn#wordn bool#bool bool#bool
                             wordn#wordn wordn#wordn wordn#wordn bool#bool
                             bool#bool wordn#wordn';;

let RstE = new_recursive_definition
    false
    r_env
    'RstE'
    "RstE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = Rst";;

let I_ad_inE = new_recursive_definition
    false
    r_env
    'I_ad_inE'
    "I_ad_inE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = I_ad_in";;

let I_rale_E = new_recursive_definition
    false
    r_env
    'I_rale_E'
    "I_rale_E (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
```

```
                ChannelID CB_parity MB_parity C_ss)
    = I_rale_";;

let I_last_E = new_recursive_definition
    false
    r_env
    'I_last_E'
    "I_last_E (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = I_last_";;

let I_be_E = new_recursive_definition
    false
    r_env
    'I_be_E'
    "I_be_E (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = I_be_";;

let I_mrdy_E = new_recursive_definition
    false
    r_env
    'I_mrdy_E'
    "I_mrdy_E (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = I_mrdy_";;

let Disable_intE = new_recursive_definition
    false
    r_env
    'Disable_intE'
    "Disable_intE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = Disable_int";;

let Disable_writesE = new_recursive_definition
    false
    r_env
    'Disable_writesE'
    "Disable_writesE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = Disable_writes";;

let Cpu_failE = new_recursive_definition
    false
    r_env
    'Cpu_failE'
    "Cpu_failE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = Cpu_fail";;

let Reset_cpuE = new_recursive_definition
    false
    r_env
    'Reset_cpuE'
    "Reset_cpuE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = Reset_cpu";;

let Piu_failE = new_recursive_definition
    false
    r_env
    'Piu_failE'
    "Piu_failE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
```

```
                ChannelID CB_parity MB_parity C_ss)
    = Piu_fail";;

let Pmm_failE = new_recursive_definition
    false
    r_env
    'Pmm_failE'
    "Pmm_failE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = Pmm_fail";;

let S_stateE = new_recursive_definition
    false
    r_env
    'S_stateE'
    "S_stateE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = S_state";;

let IdE = new_recursive_definition
    false
    r_env
    'IdE'
    "IdE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = Id";;

let ChannelIDE = new_recursive_definition
    false
    r_env
    'ChannelIDE'
    "ChannelIDE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = ChannelID";;

let CB_parityE = new_recursive_definition
    false
    r_env
    'CB_parityE'
    "CB_parityE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = CB_parity";;

let MB_parityE = new_recursive_definition
    false
    r_env
    'MB_parityE'
    "MB_parityE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = MB_parity";;

let C_ssE = new_recursive_definition
    false
    r_env
    'C_ssE'
    "C_ssE (REnv Rst I_ad_in I_rale_ I_last_ I_be_ I_mrdy_ Disable_int
                Disable_writes Cpu_fail Reset_cpu Piu_fail Pmm_fail S_state Id
                ChannelID CB_parity MB_parity C_ss)
    = C_ss";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm r_env);;

let Env_Selectors_Work = prove_thm
   ('Env_Selectors_Work',
    "!e:r_env.
```

```
    e = (REnv (RstE e) (I_ad_inE e) (I_rale_E e) (I_last_E e) (I_be_E e)
              (I_mrdy_E e) (Disable_intE e) (Disable_writesE e) (Cpu_failE e)
              (Reset_cpuE e) (Piu_failE e) (Pmm_failE e) (S_stateE e) (IdE e)
              (ChannelIDE e) (CB_parityE e) (MB_parityE e) (C_ssE e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:r_env" Env_CASES)
    THEN REWRITE_TAC [RstE; I_ad_inE; I_rale_E; I_last_E; I_be_E; I_mrdy_E;
                      Disable_intE; Disable_writesE; Cpu_failE; Reset_cpuE;
                      Piu_failE; Pmm_failE; S_stateE; IdE; ChannelIDE;
                      CB_parityE; MB_parityE; C_ssE]
  );;

%----------------------------------------------------------------------------
   Abstract data type for the output.
----------------------------------------------------------------------------%

let r_out =
   define_type 'r_out'
               'r_out = ROut busn#busn wire#wire bool#bool bool#bool bool#bool
                            bool#bool wordn#wordn wordn#wordn bool#bool
                            bool#bool';;

let I_ad_outO = new_recursive_definition
    false
    r_out
    'I_ad_outO'
    "I_ad_outO (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                     Reset_error Pmm_invalid)
    = I_ad_out";;

let I_srdy_O = new_recursive_definition
    false
    r_out
    'I_srdy_O'
    "I_srdy_O (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                    Reset_error Pmm_invalid)
    = I_srdy_";;

let Int0_O = new_recursive_definition
    false
    r_out
    'Int0_O'
    "Int0_O (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                  Reset_error Pmm_invalid)
    = Int0_";;

let Int1O = new_recursive_definition
    false
    r_out
    'Int1O'
    "Int1O (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                 Reset_error Pmm_invalid)
    = Int1";;

let Int2O = new_recursive_definition
    false
    r_out
    'Int2O'
    "Int2O (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                 Reset_error Pmm_invalid)
    = Int2";;

let Int3_O = new_recursive_definition
    false
    r_out
    'Int3_O'
    "Int3_O (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                  Reset_error Pmm_invalid)
    = Int3_";;

let CcrO = new_recursive_definition
    false
```

```
    r_out
    'CcrO'
    "CcrO (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                Reset_error Pmm_invalid)
    = Ccr";;

let LedO = new_recursive_definition
    false
    r_out
    'LedO'
    "LedO (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                Reset_error Pmm_invalid)
    = Led";;

let Reset_errorO = new_recursive_definition
    false
    r_out
    'Reset_errorO'
    "Reset_errorO (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                        Reset_error Pmm_invalid)
    = Reset_error";;

let Pmm_invalidO = new_recursive_definition
    false
    r_out
    'Pmm_invalidO'
    "Pmm_invalidO (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led
                        Reset_error Pmm_invalid)
    = Pmm_invalid";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm r_out);;

let Out_Selectors_Work = prove_thm
   ('Out_Selectors_Work',
    "!p:r_out.
     p = (ROut (I_ad_outO p) (I_srdy_O p) (Int0_O p) (Int1O p) (Int2O p)
               (Int3_O p) (CcrO p) (LedO p) (Reset_errorO p) (Pmm_invalidO p))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "p:r_out" Out_CASES)
    THEN REWRITE_TAC [I_ad_outO; I_srdy_O; Int0_O; Int1O; Int2O; Int3_O; CcrO;
                      LedO; Reset_errorO; Pmm_invalidO]
   );;

close_theory();;


%-----------------------------------------------------------------------------
   File:        rblock_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        5 March 1993

   This file contains the ml source for the gate-level specification of the
   R-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.

-----------------------------------------------------------------------------%
set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/rport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

set_flag ('timing', true);;

system 'rm rblock_def.th';;

new_theory 'rblock_def';;
```

```
loadf 'abs_theory';;
loadf 'aux_defs';;

map new_parent ['raux_def';'wordn_def';'array_def';'counters_def';'ineq'];;
map load_parent ['gates_def1';'ffs_def';'latches_def';'datapaths_def1';
                 'piuaux_def';'buses_def'];;


let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%-----------------------------------------------------------------------------
R-Port controller state machine.
-----------------------------------------------------------------------------%

let FSM_GATE = new_definition
   ('FSM_GATE',
    "! (ale_in_ mrdy_in_ last_in_ rst_in :time->bool#bool)
       (state :time->rfsm_ty)
       (ale_ mrdy_ last_ rst :time->bool)
       (s0_out s1_out cntlatch_out srdy_out_ :time->bool#bool) .
     FSM_GATE ale_in_ mrdy_in_ last_in_ rst_in
              ale_ mrdy_ last_ rst state
              s0_out s1_out cntlatch_out srdy_out_ =
     !(t:time).
        (state (t+1) =
           (rst t) => RI |
           ((state t) = RI) => ((~ale_ t) => RA | RI) |
           ((state t) = RA) => ((~mrdy_ t) => RD | RA) |
           (~last_ t) => RI | RA) /\
        (ale_ (t+1) = BSel(ale_in_ t)) /\
        (mrdy_ (t+1) = BSel(mrdy_in_ t)) /\
        (last_ (t+1) = BSel(last_in_ t)) /\
        (rst (t+1) = BSel(rst_in t)) /\
        (s0_out t = ((state (t+1) = RD), (state (t+1) = RD))) /\
        (s1_out t =
           (((state (t+1) = RA) \/ (state (t+1) = RD)),
            ((state (t+1) = RA) \/ (state (t+1) = RD)))) /\
        (cntlatch_out t =
           (((state t = RI) /\ ~ale_ t), ((state t = RI) /\ ~ale_ t))) /\
        (srdy_out_ t =
           ((~((state t = RA) /\ ~mrdy_ t)), (~((state t = RA) /\ ~mrdy_ t))))"
   );;

%-----------------------------------------------------------------------------
   R_wr latch definition.
-----------------------------------------------------------------------------%

let Wr_Lat_GATE = new_definition
   ('Wr_Lat_GATE',
    "! (iad_in :time->wordn#wordn)
       (wr_inE wr_outQ :time->bool#bool)
       (r_wr :time->bool) .
     Wr_Lat_GATE iad_in wr_inE r_wr wr_outQ =
     !(t:time).
        (r_wr (t+1) =
           (BSel(wr_inE t)) => (ELEMENT (BSel(iad_in t)) (27)) | r_wr t) /\
        (wr_outQ t = (r_wr t, r_wr (t+1)))"
   );;

%-----------------------------------------------------------------------------
   Generation logic for control signals dp_read, r_write, r_read, icr_rd_en,
   srdy_en.
-----------------------------------------------------------------------------%

let RW_Sigs_GATE = new_definition
   ('RW_Sigs_GATE',
    "! (r_wr s0 s1 disable_writes dp_read r_write r_read :time->bool#bool)
       (icr_rd_en srdy_en :time->bool#bool) .
     RW_Sigs_GATE r_wr s0 s1 disable_writes dp_read r_write r_read icr_rd_en
                  srdy_en =
     !(t:time).
        (dp_read t =
```

```
          (((~ASel(r_wr t)) /\ (ASel(s0 t) \/ (ASel(s1 t)))),
           ((~BSel(r_wr t)) /\ (BSel(s0 t) \/ (BSel(s1 t))))))) /\
       (r_write t =
          ((~ASel(disable_writes t) /\ ASel(r_wr t) /\ ASel(s0 t) /\
            ASel(s1 t)),
           (~BSel(disable_writes t) /\ BSel(r_wr t) /\ BSel(s0 t) /\
            BSel(s1 t)))) /\
       (r_read t =
          ((~ASel(r_wr t) /\ ~ASel(s0 t) /\ ASel(s1 t)),
           (~BSel(r_wr t) /\ ~BSel(s0 t) /\ BSel(s1 t)))) /\
       (icr_rd_en t =
          ((~ASel(s0 t) /\ ASel(s1 t)),
           (~BSel(s0 t) /\ BSel(s1 t)))) /\
       (srdy_en t =
          ((ASel(s0 t) \/ ASel(s1 t)),
           (BSel(s0 t) \/ BSel(s1 t))))"
  );;

%-----------------------------------------------------------------------------
   R_reg_sel counter and logic.
-----------------------------------------------------------------------------%

let Reg_Sel_Ctr_GATE = new_definition
   ('Reg_Sel_Ctr_GATE',
    "! (iad_in outQ :time->wordn#wordn)
       (inL inU_ :time->bool#bool)
       (r_reg_sel :time->wordn) .
     Reg_Sel_Ctr_GATE iad_in inL inU_ r_reg_sel outQ =
     !(t:time).
        (r_reg_sel (t+1) =
           (BSel(inL t)) => SUBARRAY (BSel(iad_in t)) (3,0) |
           (~BSel(inU_ t)) => INCN 3 (r_reg_sel t) | r_reg_sel t) /\
        (outQ t =
           (((~ASel(inU_ t)) => INCN 3 (r_reg_sel t) | r_reg_sel t),
            ((~BSel(inU_ t)) => INCN 3 (r_reg_sel t) | r_reg_sel t)))"
  );;

%-----------------------------------------------------------------------------
   Generation logic for register file control signals.
-----------------------------------------------------------------------------%

let Reg_File_Ctl_GATE = new_definition
   ('Reg_File_Ctl_GATE',
    "! (reg_sel :time->wordn#wordn)
       (write read icr_rd_en cir_wr01 cir_wr23 c0ir_wr c0ir_rd :time->bool#bool)
       (c0or_rd c1ir_wr c1ir_rd c1or_rd c2ir_wr c2ir_rd c2or_rd :time->bool#bool)
       (c3ir_wr c3ir_rd c3or_rd icr_wr_feedback icr_select icr_rd :time->bool#bool)
       (ccr_wr ccr_rd gcr_wr gcr_rd sr_rd :time->bool#bool) .
     Reg_File_Ctl_GATE reg_sel write read icr_rd_en
                       cir_wr01 cir_wr23
                       c0ir_wr c0ir_rd c0or_rd c1ir_wr c1ir_rd c1or_rd
                       c2ir_wr c2ir_rd c2or_rd c3ir_wr c3ir_rd c3or_rd
                       icr_wr_feedback icr_select icr_rd
                       ccr_wr ccr_rd gcr_wr gcr_rd sr_rd =
     !(t:time).
        (cir_wr01 t =
           ((ASel(write t) /\ ((ASel(reg_sel t) = WORDN 3 8) \/
                               (ASel(reg_sel t) = WORDN 3 9))),
            (BSel(write t) /\ ((BSel(reg_sel t) = WORDN 3 8) \/
                               (BSel(reg_sel t) = WORDN 3 9))))) /\
        (cir_wr23 t =
           ((ASel(write t) /\ ((ASel(reg_sel t) = WORDN 3 10) \/
                               (ASel(reg_sel t) = WORDN 3 11))),
            (BSel(write t) /\ ((BSel(reg_sel t) = WORDN 3 10) \/
                               (BSel(reg_sel t) = WORDN 3 11))))) /\
        (c0ir_wr t =
           ((ASel(write t) /\ (ASel(reg_sel t) = WORDN 3 8)),
            (BSel(write t) /\ (BSel(reg_sel t) = WORDN 3 8)))) /\
        (c0ir_rd t =
           ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 8)),
            (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 8)))) /\
        (c0or_rd t =
```

```
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 12)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 12)))) /\
        (c1ir_wr t =
          ((ASel(write t) /\ (ASel(reg_sel t) = WORDN 3 9)),
           (BSel(write t) /\ (BSel(reg_sel t) = WORDN 3 9)))) /\
        (c1ir_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 9)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 9)))) /\
        (c1or_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 13)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 13)))) /\
        (c2ir_wr t =
          ((ASel(write t) /\ (ASel(reg_sel t) = WORDN 3 10)),
           (BSel(write t) /\ (BSel(reg_sel t) = WORDN 3 10)))) /\
        (c2ir_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 10)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 10)))) /\
        (c2or_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 14)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 14)))) /\
        (c3ir_wr t =
          ((ASel(write t) /\ (ASel(reg_sel t) = WORDN 3 11)),
           (BSel(write t) /\ (BSel(reg_sel t) = WORDN 3 11)))) /\
        (c3ir_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 11)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 11)))) /\
        (c3or_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 15)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 15)))) /\
        (icr_wr_feedback t =
          ((ASel(write t) /\ ((ASel(reg_sel t) = WORDN 3 0) \/
                              (ASel(reg_sel t) = WORDN 3 1))),
           (BSel(write t) /\ ((BSel(reg_sel t) = WORDN 3 0) \/
                              (BSel(reg_sel t) = WORDN 3 1))))) /\
        (icr_select t =
          ((~(ASel(reg_sel t) = WORDN 3 1)),
           (~(BSel(reg_sel t) = WORDN 3 1)))) /\
        (icr_rd t =
          ((ASel(icr_rd_en t) /\ ((ASel(reg_sel t) = WORDN 3 0) \/
                                  (ASel(reg_sel t) = WORDN 3 1))),
           (BSel(icr_rd_en t) /\ ((BSel(reg_sel t) = WORDN 3 0) \/
                                  (BSel(reg_sel t) = WORDN 3 1))))) /\
        (ccr_wr t =
          ((ASel(write t) /\ (ASel(reg_sel t) = WORDN 3 3)),
           (BSel(write t) /\ (BSel(reg_sel t) = WORDN 3 3)))) /\
        (ccr_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 3)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 3)))) /\
        (gcr_wr t =
          ((ASel(write t) /\ (ASel(reg_sel t) = WORDN 3 2)),
           (BSel(write t) /\ (BSel(reg_sel t) = WORDN 3 2)))) /\
        (gcr_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 2)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 2)))) /\
        (sr_rd t =
          ((ASel(read t) /\ (ASel(reg_sel t) = WORDN 3 4)),
           (BSel(read t) /\ (BSel(reg_sel t) = WORDN 3 4))))"
   );;

%----------------------------------------------------------------------------
   Input logic for R_int1_en, R_int2_en latches.
----------------------------------------------------------------------------%

let Ctr_Int_Logic_GATE = new_definition
   ('Ctr_Int_Logic_GATE',
    "! (one_shot interrupt reload cout cout_del cir_wr :time->bool#bool)
       (cout_out int_en_inR int_en_inS int_en_inE c_ld :time->bool#bool) .
     Ctr_Int_Logic_GATE one_shot interrupt reload cout cout_del cir_wr
                        cout_out int_en_inR int_en_inS int_en_inE c_ld =
     !(t:time).
        (cout_out t = ((ASel(cout t)),ASel(cout t))) /\
        (int_en_inR t =
```

```
          (((ASel(one_shot t) /\ ASel(cout_del t)) \/ ~ASel(interrupt t)),
           ((BSel(one_shot t) /\ BSel(cout_del t)) \/ ~BSel(interrupt t)))) /\
       (int_en_inS t =
          ((ASel(interrupt t) /\ ((ASel(cout t) /\ ASel(reload t)) \/
                                  ASel(cir_wr t))),
           (BSel(interrupt t) /\ ((ASel(cout t) /\ BSel(reload t)) \/
                                  BSel(cir_wr t))))) /\
       (int_en_inE t =
          ((ASel(int_en_inR t) \/ ASel(int_en_inS t)),
           (BSel(int_en_inR t) \/ BSel(int_en_inS t)))) /\
       (c_ld t =
           (((ASel(cout t) /\ ASel(reload t)) \/ ASel(cir_wr t)),
            ((ASel(cout t) /\ BSel(reload t)) \/ BSel(cir_wr t))))"
  );;

%-----------------------------------------------------------------------------
   Input logic for R_int0_en, R_int3_en latches.
-----------------------------------------------------------------------------%

let And_Tree_GATE = new_definition
   ('And_Tree_GATE',
    "! (icr :time->wordn#wordn)
       (out0 out3 :time->bool#bool) .
     And_Tree_GATE icr out0 out3 =
     !(t:time).
        (out0 t =
           (((ELEMENT (ASel(icr t)) (0)) /\ (ELEMENT (ASel(icr t)) (8)) \/
             (ELEMENT (ASel(icr t)) (1)) /\ (ELEMENT (ASel(icr t)) (9)) \/
             (ELEMENT (ASel(icr t)) (2)) /\ (ELEMENT (ASel(icr t)) (10)) \/
             (ELEMENT (ASel(icr t)) (3)) /\ (ELEMENT (ASel(icr t)) (11)) \/
             (ELEMENT (ASel(icr t)) (4)) /\ (ELEMENT (ASel(icr t)) (12)) \/
             (ELEMENT (ASel(icr t)) (5)) /\ (ELEMENT (ASel(icr t)) (13)) \/
             (ELEMENT (ASel(icr t)) (6)) /\ (ELEMENT (ASel(icr t)) (14)) \/
             (ELEMENT (ASel(icr t)) (7)) /\ (ELEMENT (ASel(icr t)) (15))),
            ((ELEMENT (ASel(icr t)) (0)) /\ (ELEMENT (ASel(icr t)) (8)) \/
             (ELEMENT (ASel(icr t)) (1)) /\ (ELEMENT (ASel(icr t)) (9)) \/
             (ELEMENT (ASel(icr t)) (2)) /\ (ELEMENT (ASel(icr t)) (10)) \/
             (ELEMENT (ASel(icr t)) (3)) /\ (ELEMENT (ASel(icr t)) (11)) \/
             (ELEMENT (ASel(icr t)) (4)) /\ (ELEMENT (ASel(icr t)) (12)) \/
             (ELEMENT (ASel(icr t)) (5)) /\ (ELEMENT (ASel(icr t)) (13)) \/
             (ELEMENT (ASel(icr t)) (6)) /\ (ELEMENT (ASel(icr t)) (14)) \/
             (ELEMENT (ASel(icr t)) (7)) /\ (ELEMENT (ASel(icr t)) (15))))) /\
        (out3 t =
           (((ELEMENT (ASel(icr t)) (16)) /\ (ELEMENT (ASel(icr t)) (24)) \/
             (ELEMENT (ASel(icr t)) (17)) /\ (ELEMENT (ASel(icr t)) (25)) \/
             (ELEMENT (ASel(icr t)) (18)) /\ (ELEMENT (ASel(icr t)) (26)) \/
             (ELEMENT (ASel(icr t)) (19)) /\ (ELEMENT (ASel(icr t)) (27)) \/
             (ELEMENT (ASel(icr t)) (20)) /\ (ELEMENT (ASel(icr t)) (28)) \/
             (ELEMENT (ASel(icr t)) (21)) /\ (ELEMENT (ASel(icr t)) (29)) \/
             (ELEMENT (ASel(icr t)) (22)) /\ (ELEMENT (ASel(icr t)) (30)) \/
             (ELEMENT (ASel(icr t)) (23)) /\ (ELEMENT (ASel(icr t)) (31))),
            ((ELEMENT (ASel(icr t)) (16)) /\ (ELEMENT (ASel(icr t)) (24)) \/
             (ELEMENT (ASel(icr t)) (17)) /\ (ELEMENT (ASel(icr t)) (25)) \/
             (ELEMENT (ASel(icr t)) (18)) /\ (ELEMENT (ASel(icr t)) (26)) \/
             (ELEMENT (ASel(icr t)) (19)) /\ (ELEMENT (ASel(icr t)) (27)) \/
             (ELEMENT (ASel(icr t)) (20)) /\ (ELEMENT (ASel(icr t)) (28)) \/
             (ELEMENT (ASel(icr t)) (21)) /\ (ELEMENT (ASel(icr t)) (29)) \/
             (ELEMENT (ASel(icr t)) (22)) /\ (ELEMENT (ASel(icr t)) (30)) \/
             (ELEMENT (ASel(icr t)) (23)) /\ (ELEMENT (ASel(icr t)) (31)))))"
  );;

%-----------------------------------------------------------------------------
   Generation logic for Int0_, Int3_ signals.
-----------------------------------------------------------------------------%

let Reg_Int_Logic_GATE = new_definition
   ('Reg_Int_Logic_GATE',
    "! (int0_en int0_dis int3_en int3_dis disable_int int0_
        int3_ :time->bool#bool).
     Reg_Int_Logic_GATE int0_en int0_dis int3_en int3_dis disable_int int0_
                        int3_ =
     !(t:time).
```

```
        (int0_ t =
          ((~(ASel(int0_en t) /\ ~ASel(int0_dis t) /\ ~ASel(disable_int t))),
           (~(BSel(int0_en t) /\ ~BSel(int0_dis t) /\ ~BSel(disable_int t))))) /\
        (int3_ t =
          ((~(ASel(int3_en t) /\ ~ASel(int3_dis t) /\ ~ASel(disable_int t))),
           (~(BSel(int3_en t) /\ ~BSel(int3_dis t) /\ ~BSel(disable_int t)))))"
   );;

%-----------------------------------------------------------------------------
   Virtual logic to package several R-Port inputs into single SR input word.
-----------------------------------------------------------------------------%

let SR_Inputs_GATE = new_definition
   ('SR_Inputs_GATE',
    "! (cpu_fail reset_cpu s_state id channelID c_ss sr_inp :time->wordn#wordn)
       (piu_fail pmm_fail cb_parity mb_parity :time->bool#bool) .
     SR_Inputs_GATE cpu_fail reset_cpu piu_fail pmm_fail s_state
                    id channelID cb_parity c_ss mb_parity sr_inp =
     !(t:time).
        let a1 = (MALTER ARBN (1,0) (ASel(cpu_fail t))) in
        let a3 = (MALTER a1 (3,2) (ASel(reset_cpu t))) in
        let a5 = (ALTER a3 (8) (ASel(piu_fail t))) in
        let a6 = (ALTER a5 (9) (ASel(pmm_fail t))) in
        let a7 = (MALTER a6 (15,12) (ASel(s_state t))) in
        let a8 = (MALTER a7 (21,16) (ASel(id t))) in
        let a9 = (MALTER a8 (23,22) (ASel(channelID t))) in
        let a10 = (ALTER a9 (24) (ASel(cb_parity t))) in
        let a11 = (MALTER a10 (27,25) (ASel(c_ss t))) in
        let a12 = (ALTER a11 (28) (ASel(mb_parity t))) in
        let b1 = (MALTER ARBN (1,0) (BSel(cpu_fail t))) in
        let b3 = (MALTER b1 (3,2) (BSel(reset_cpu t))) in
        let b5 = (ALTER b3 (8) (BSel(piu_fail t))) in
        let b6 = (ALTER b5 (9) (BSel(pmm_fail t))) in
        let b7 = (MALTER b6 (15,12) (BSel(s_state t))) in
        let b8 = (MALTER b7 (21,16) (BSel(id t))) in
        let b9 = (MALTER b8 (23,22) (BSel(channelID t))) in
        let b10 = (ALTER b9 (24) (BSel(cb_parity t))) in
        let b11 = (MALTER b10 (27,25) (BSel(c_ss t))) in
        let b12 = (ALTER b11 (28) (BSel(mb_parity t))) in
        (sr_inp t = (a12, b12))"
   );;

%-----------------------------------------------------------------------------
   Virtual logic to distribute single GCR output word as several pieces.
-----------------------------------------------------------------------------%

let GCR_Outputs_GATE = new_definition
   ('GCR_Outputs_GATE',
    "! (gcr_out led :time->wordn#wordn)
       (reload01 oneshot01 interrupt01 enable01 reload23 :time->bool#bool)
       (oneshot23 interrupt23 enable23 reset_error pmm_invalid :time->bool#bool).
     GCR_Outputs_GATE gcr_out led reload01 oneshot01 interrupt01
                      enable01 reload23 oneshot23 interrupt23 enable23
                      reset_error pmm_invalid =
     !(t:time).
        (led t =
           ((SUBARRAY (ASel(gcr_out t)) (3,0)),
            (SUBARRAY (BSel(gcr_out t)) (3,0)))) /\
        (reload01 t =
           ((ELEMENT (ASel(gcr_out t)) (16)),
            (ELEMENT (BSel(gcr_out t)) (16)))) /\
        (oneshot01 t =
           ((ELEMENT (ASel(gcr_out t)) (17)),
            (ELEMENT (BSel(gcr_out t)) (17)))) /\
        (interrupt01 t =
           ((ELEMENT (ASel(gcr_out t)) (18)),
            (ELEMENT (BSel(gcr_out t)) (18)))) /\
        (enable01 t =
           ((ELEMENT (ASel(gcr_out t)) (19)),
            (ELEMENT (BSel(gcr_out t)) (19)))) /\
        (reload23 t =
           ((ELEMENT (ASel(gcr_out t)) (20)),
```

```
            (ELEMENT (BSel(gcr_out t)) (20)))) /\
        (oneshot23 t =
           ((ELEMENT (ASel(gcr_out t)) (21)),
            (ELEMENT (BSel(gcr_out t)) (21)))) /\
        (interrupt23 t =
           ((ELEMENT (ASel(gcr_out t)) (22)),
            (ELEMENT (BSel(gcr_out t)) (22)))) /\
        (enable23 t =
           ((ELEMENT (ASel(gcr_out t)) (23)),
            (ELEMENT (BSel(gcr_out t)) (23)))) /\
        (reset_error t =
           ((ELEMENT (ASel(gcr_out t)) (24)),
            (ELEMENT (BSel(gcr_out t)) (24)))) /\
        (pmm_invalid t =
           ((ELEMENT (ASel(gcr_out t)) (28)),
            (ELEMENT (BSel(gcr_out t)) (28))))"
   );;

%-----------------------------------------------------------------------------
R-Port block.
-----------------------------------------------------------------------------%

let RBlock_GATE = new_definition
('RBlock_GATE',
 "! (rep :^REP_ty) (s :time->r_state) (e :time->r_env) (p :time->r_out) .
     RBlock_GATE rep s e p =
? (fsm_s0 fsm_s1 fsm_cntlatch fsm_srdy_ srdy_en wr_inE :time->bool#bool)
        (wr_outQ dp_read r_write r_read icr_rd_en srdy_en :time->bool#bool)
        (c13or_ld srdy_del_outQ_ :time->bool#bool)
        (reg_sel icr_out BusA busA_latch_out BusB_in ccr_out :time->wordn#wordn)
        (gcr_out sr_inp :time->wordn#wordn)
        (r_cir_wr01 r_cir_wr23 c0ir_wr c0ir_rd c0or_rd c1ir_wr :time->bool#bool)
        (c1ir_rd c1or_rd c2ir_wr c2ir_rd c2or_rd c3ir_wr :time->bool#bool)
        (c3ir_rd c3or_rd icr_wr_feedback icr_select icr_rd :time->bool#bool)
        (ccr_wr ccr_rd gcr_wr gcr_rd sr_rd icr_ld c01_cout :time->bool#bool)
        (c01_coutA c23_cout c23_coutA c01_cout_del_outQ :time->bool#bool)
        (c23_cout_del_outQ oneshot01 interrupt01 reload01 :time->bool#bool)
        (int1_en_inR int1_en_inS int1_en_inE c01_ld oneshot23 :time->bool#bool)
        (interrupt23 reload23 int2_en_inR int2_en_inS :time->bool#bool)
        (int2_en_inE c23_ld int1_en_outQ int2_en_outQ :time->bool#bool)
        (disable_int_ int0_en int3_en :time->bool#bool)
        (int0_dis_outQ int3_dis_outQ :time->bool#bool)
        (enable01 c0_cout c2_cout enable23 :time->bool#bool)
        (BusA_c0_out1 BusA_c0_out2 BusA_c1_out1 BusA_c1_out2 :time->busn#busn)
        (BusA_c2_out1 BusA_c2_out2 BusA_c3_out1 BusA_c3_out2 :time->busn#busn)
        (BusA_icr_out BusA_ccr_out BusA_gcr_out BusA_sr_out :time->busn#busn) .

(FSM_GATE (sig I_rale_E e) (sig I_mrdy_E e) (sig I_last_E e) (sig RstE e)
               (sig R_fsm_ale_S s) (sig R_fsm_mrdy_S s) (sig R_fsm_last_S s)
               (sig R_fsm_rstS s) (sig R_fsm_stateS s)
 fsm_s0 fsm_s1 fsm_cntlatch fsm_srdy_) /\
(TRIBUF_GATE fsm_srdy_ srdy_en (sig I_srdy_O p)) /\
(NOT_GATE (sig I_rale_E e) wr_inE) /\
   (Wr_Lat_GATE (sig I_ad_inE e) wr_inE (sig R_wrS s) wr_outQ) /\
(RW_Sigs_GATE wr_outQ fsm_s0 fsm_s1 (sig Disable_writesE e) dp_read
                    r_write r_read icr_rd_en srdy_en) /\
(DFFA_GATE fsm_cntlatch  (sig R_cntlatch_delS s) c13or_ld) /\
(DFFA_GATE fsm_srdy_ (sig R_srdy_del_S s) srdy_del_outQ_) /\
   (Reg_Sel_Ctr_GATE (sig I_ad_inE e) wr_inE srdy_del_outQ_
                     (sig R_reg_selS s) reg_sel) /\
(Reg_File_Ctl_GATE reg_sel r_write r_read icr_rd_en
 r_cir_wr01 r_cir_wr23
 c0ir_wr c0ir_rd c0or_rd c1ir_wr c1ir_rd c1or_rd
 c2ir_wr c2ir_rd c2or_rd c3ir_wr c3ir_rd c3or_rd
  icr_wr_feedback icr_select icr_rd
  ccr_wr ccr_rd gcr_wr gcr_rd sr_rd) /\
(DFFA_GATE icr_wr_feedback (sig R_icr_loadS s) icr_ld) /\
(DFFA_GATE c01_coutA (sig R_c01_cout_delS s) c01_cout_del_outQ) /\
(DFFA_GATE c23_coutA (sig R_c23_cout_delS s) c23_cout_del_outQ) /\
(Ctr_Int_Logic_GATE oneshot01 interrupt01 reload01 c01_cout
                        c01_cout_del_outQ r_cir_wr01 c01_coutA int1_en_inR
                        int1_en_inS int1_en_inE c01_ld) /\
```

```
(Ctr_Int_Logic_GATE oneshot23 interrupt23 reload23 c23_cout
                        c23_cout_del_outQ r_cir_wr23 c23_coutA int2_en_inR
                        int2_en_inS int2_en_inE c23_ld) /\
(DSRELatB_GATE GND int1_en_inS int1_en_inR int1_en_inE (sig R_int1_enS s)
                       int1_en_outQ) /\
(DSRELatB_GATE GND int2_en_inS int2_en_inR int2_en_inE (sig R_int2_enS s)
                       int2_en_outQ) /\
(NOT_GATE (sig Disable_intE e) disable_int_) /\
(AND3_GATE c01_coutA int1_en_outQ disable_int_ (sig Int1O p)) /\
(AND3_GATE c23_coutA int2_en_outQ disable_int_ (sig Int2O p)) /\
(And_Tree_GATE icr_out int0_en int3_en) /\

(DFFA_GATE int0_en (sig R_int0_disS s) int0_dis_outQ) /\
(DFFA_GATE int3_en (sig R_int3_disS s) int3_dis_outQ) /\
(Reg_Int_Logic_GATE int0_en int0_dis_outQ int3_en int3_dis_outQ
  (sig Disable_intE e) (sig Int0_O p) (sig Int3_O p)) /\
(DLatNA_GATE BusA (sig R_busA_latchS s) busA_latch_out) /\
(TRIBUFn_GATE busA_latch_out dp_read (sig I_ad_outO p)) /\
(BUF_GATE (sig I_ad_inE e) BusB_in) /\
(DP_CTR_GATE BusB_in c0ir_wr c01_ld c0ir_rd enable01 VDD fsm_cntlatch
 c0or_rd (sig R_ctr0_inS s) (sig R_ctr0_mux_selS s)
                   (sig R_ctr0_irdenS s) (sig R_ctr0S s)
                   (sig R_ctr0_cryS s) (sig R_ctr0_newS s)
                   (sig R_ctr0_outS s) (sig R_ctr0_ordenS s)
 BusA_c0_out1 BusA_c0_out2 c0_cout) /\
(DP_CTR_GATE BusB_in c1ir_wr c01_ld c1ir_rd VDD c0_cout c13or_ld
 c1or_rd (sig R_ctr1_inS s) (sig R_ctr1_mux_selS s)
                   (sig R_ctr1_irdenS s) (sig R_ctr1S s)
                   (sig R_ctr1_cryS s) (sig R_ctr1_newS s)
                   (sig R_ctr1_outS s) (sig R_ctr1_ordenS s)
 BusA_c1_out1 BusA_c1_out2 c01_cout) /\
(DP_CTR_GATE BusB_in c2ir_wr c23_ld c2ir_rd enable23 VDD fsm_cntlatch
 c2or_rd (sig R_ctr2_inS s) (sig R_ctr2_mux_selS s)
                   (sig R_ctr2_irdenS s) (sig R_ctr2S s)
                   (sig R_ctr2_cryS s) (sig R_ctr2_newS s)
                   (sig R_ctr2_outS s) (sig R_ctr2_ordenS s)
 BusA_c2_out1 BusA_c2_out2 c2_cout) /\
(DP_CTR_GATE BusB_in c3ir_wr c23_ld c3ir_rd VDD c2_cout c13or_ld
 c3or_rd (sig R_ctr3_inS s) (sig R_ctr3_mux_selS s)
                   (sig R_ctr3_irdenS s) (sig R_ctr3S s)
                   (sig R_ctr3_cryS s) (sig R_ctr3_newS s)
                   (sig R_ctr3_outS s) (sig R_ctr3_ordenS s)
 BusA_c3_out1 BusA_c3_out2 c23_cout) /\
(DP_ICR_GATE rep BusA BusB_in icr_wr_feedback icr_wr_feedback icr_select
                   icr_ld icr_rd (sig R_icr_oldS s) (sig R_icr_maskS s)
                   (sig R_icrS s) (sig R_icr_rdenS s) BusA_icr_out icr_out) /\
(DP_CR_GATE BusB_in ccr_wr ccr_rd (sig R_ccrS s) (sig R_ccr_rdenS s)
                  BusA_ccr_out (sig CcrO p)) /\
(DP_CR_GATE BusB_in gcr_wr gcr_rd (sig R_gcrS s) (sig R_gcr_rdenS s)
                  BusA_gcr_out gcr_out) /\
   (GCR_Outputs_GATE gcr_out (sig LedO p) reload01 oneshot01 interrupt01
                        enable01 reload23 oneshot23 interrupt23 enable23
                        (sig Reset_errorO p) (sig Pmm_invalidO p)) /\
   (SR_Inputs_GATE (sig Cpu_failE e) (sig Reset_cpuE e) (sig Piu_failE e)
                      (sig Pmm_failE e) (sig S_stateE e) (sig IdE e)
                      (sig ChannelIDE e) (sig CB_parityE e) (sig C_ssE e)
                      (sig MB_parityE e) sr_inp) /\
(DP_SR_GATE sr_inp fsm_cntlatch sr_rd (sig R_srS s) (sig R_sr_rdenS s)
                 BusA_sr_out) /\
      (JOIN12n_GATE (31,0)
                    BusA_c0_out1 BusA_c0_out2 BusA_c1_out1 BusA_c1_out2
                    BusA_c2_out1 BusA_c2_out2 BusA_c3_out1 BusA_c3_out2
                    BusA_icr_out BusA_ccr_out BusA_gcr_out BusA_sr_out BusA)"
);;

let RBlock_EXP = save_thm
   ('RBlock_EXP',
    (BETA_RULE
     (REWRITE_RULE
       [FSM_GATE;Wr_Lat_GATE;RW_Sigs_GATE;Reg_Sel_Ctr_GATE; Reg_File_Ctl_GATE;
        Ctr_Int_Logic_GATE;And_Tree_GATE; Reg_Int_Logic_GATE;
        (EXPAND_LET_RULE SR_Inputs_GATE); GCR_Outputs_GATE;
```

```
        TRIBUF_GATE;NOT_GATE;DFFA_GATE; DLatA_GATE;DSRELatB_GATE;AND3_GATE;
        DLatNA_GATE;TRIBUFn_GATE; BUF_GATE;DP_CTR_GATE;DP_ICR_GATE;DP_CR_GATE;
        DP_SR_GATE;(EXPAND_LET_RULE JOIN12n_GATE);ASel;BSel;GND;VDD;sig]
       (SPEC_ALL RBlock_GATE)))
   );;

close_theory();;


%----------------------------------------------------------------------------

   File:        rclock_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        5 March 1993

   This file contains the ml source for the clock-level specification of the
   R-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.  The bulk of this code was
   translated from an M-language simulation program using a translator written
   by P.J. Windley at the University of Idaho.

----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm rclock_def.th';;

new_theory 'rclock_def';;

loadf 'abs_theory';;

map new_parent ['piuaux_def';'raux_def';'array_def';'wordn_def';'ineq'];;

new_type_abbrev ('timeC',":num");;

let ASel = definition 'piuaux_def' 'ASel';;
let BSel = definition 'piuaux_def' 'BSel';;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%----------------------------------------------------------------------------
Next-state definition for R-Port instruction.
----------------------------------------------------------------------------%

let RClockNSF = new_definition
    ('RClockNSF',
  "! (rep :^REP_ty) (s :r_state) (e :r_env) .
      RClockNSF rep s e =

      let R_fsm_state = R_fsm_stateS s and
          R_fsm_ale_ = R_fsm_ale_S s and
          R_fsm_mrdy_ = R_fsm_mrdy_S s and
          R_fsm_last_ = R_fsm_last_S s and
          R_fsm_rst = R_fsm_rstS s and
          R_ctr0_in = R_ctr0_inS s and
          R_ctr0_mux_sel = R_ctr0_mux_selS s and
          R_ctr0 = R_ctr0S s and
          R_ctr0_irden = R_ctr0_irdenS s and
          R_ctr0_new = R_ctr0_newS s and
          R_ctr0_cry = R_ctr0_cryS s and
          R_ctr0_out = R_ctr0_outS s and
          R_ctr0_orden = R_ctr0_ordenS s and
          R_ctr1_in = R_ctr1_inS s and
          R_ctr1_mux_sel = R_ctr1_mux_selS s and
          R_ctr1 = R_ctr1S s and
          R_ctr1_irden = R_ctr1_irdenS s and
          R_ctr1_new = R_ctr1_newS s and
```

```
        R_ctr1_cry = R_ctr1_cryS s and
        R_ctr1_out = R_ctr1_outS s and
        R_ctr1_orden = R_ctr1_ordenS s and
        R_ctr2_in = R_ctr2_inS s and
        R_ctr2_mux_sel = R_ctr2_mux_selS s and
        R_ctr2 = R_ctr2S s and
        R_ctr2_irden = R_ctr2_irdenS s and
        R_ctr2_new = R_ctr2_newS s and
        R_ctr2_cry = R_ctr2_cryS s and
        R_ctr2_out = R_ctr2_outS s and
        R_ctr2_orden = R_ctr2_ordenS s and
        R_ctr3_in = R_ctr3_inS s and
        R_ctr3_mux_sel = R_ctr3_mux_selS s and
        R_ctr3 = R_ctr3S s and
        R_ctr3_irden = R_ctr3_irdenS s and
        R_ctr3_new = R_ctr3_newS s and
        R_ctr3_cry = R_ctr3_cryS s and
        R_ctr3_out = R_ctr3_outS s and
        R_ctr3_orden = R_ctr3_ordenS s and
        R_icr_load = R_icr_loadS s and
        R_icr_old = R_icr_oldS s and
        R_icr_mask = R_icr_maskS s and
        R_icr_rden = R_icr_rdenS s and
        R_icr = R_icrS s and
        R_ccr = R_ccrS s and
        R_ccr_rden = R_ccr_rdenS s and
        R_gcr = R_gcrS s and
        R_gcr_rden = R_gcr_rdenS s and
        R_sr = R_srS s and
        R_sr_rden = R_sr_rdenS s and
        R_int0_dis = R_int0_disS s and
        R_int3_dis = R_int3_disS s and
        R_c01_cout_del = R_c01_cout_delS s and
        R_int1_en = R_int1_enS s and
        R_c23_cout_del = R_c23_cout_delS s and
        R_int2_en = R_int2_enS s and
        R_wr = R_wrS s and
        R_cntlatch_del = R_cntlatch_delS s and
        R_srdy_del_ = R_srdy_del_S s and
        R_reg_sel = R_reg_selS s and
        R_busA_latch = R_busA_latchS s in
    let Rst = RstE e and
        I_ad_in = I_ad_inE e and
        I_rale_ = I_rale_E e and
        I_last_ = I_last_E e and
        I_be_ = I_be_E e and
        I_mrdy_ = I_mrdy_E e and
        Disable_int = Disable_intE e and
        Disable_writes = Disable_writesE e and
        Cpu_fail = Cpu_failE e and
        Reset_cpu = Reset_cpuE e and
        Piu_fail = Piu_failE e and
        Pmm_fail = Pmm_failE e and
        S_state = S_stateE e and
        Id = IdE e and
        ChannelID = ChannelIDE e and
        CB_parity = CB_parityE e and
        MB_parity = MB_parityE e and
        C_ss = C_ssE e in

  let new_R_fsm_state =
((R_fsm_rst) => RI |
  ((R_fsm_state = RI) => ((~R_fsm_ale_) => RA | RI) |
  ((R_fsm_state = RA) => ((~R_fsm_mrdy_) => RD | RA) |
  ((~R_fsm_last_) => RI | RA)))) in
  let r_fsm_cntlatch = ((R_fsm_state = RI) /\ ~R_fsm_ale_) in
  let r_fsm_srdy_ = ~((R_fsm_state = RA) /\ ~R_fsm_mrdy_) in
  let new_R_wr = ((~BSel(I_rale_)) => (ELEMENT (BSel(I_ad_in)) (27)) | R_wr)
      in
  let new_R_cntlatch_del = r_fsm_cntlatch in
  let new_R_srdy_del_ = r_fsm_srdy_ in
  let new_R_reg_sel =
```

```
((~BSel(I_rale_)) => (SUBARRAY (BSel(I_ad_in)) (3,0)) |
 ((~R_srdy_del_) => (INCN 3 R_reg_sel) | R_reg_sel)) in
  let r_reg_sel = ((~R_srdy_del_) => (INCN 3 R_reg_sel) | R_reg_sel) in
  let r_writeA = (~ASel(Disable_writes) /\ R_wr /\ (new_R_fsm_state = RD)) in
  let r_writeB = (~BSel(Disable_writes) /\ new_R_wr /\
                    (new_R_fsm_state = RD)) in
  let r_readA = (~R_wr /\ (new_R_fsm_state = RA)) in
  let r_readB = (~new_R_wr /\ (new_R_fsm_state = RA)) in

  let r_cir_wr01A = ((r_writeA /\ ((r_reg_sel = (WORDN 3 8)) \/
                                   (r_reg_sel = (WORDN 3 9))))) in
  let r_cir_wr01B = ((r_writeB /\ ((r_reg_sel = (WORDN 3 8)) \/
                                   (r_reg_sel = (WORDN 3 9))))) in
  let r_cir_wr23A = ((r_writeA /\ ((r_reg_sel = (WORDN 3 10)) \/
                                   (r_reg_sel = (WORDN 3 11))))) in
  let r_cir_wr23B = ((r_writeB /\ ((r_reg_sel = (WORDN 3 10)) \/
                                   (r_reg_sel = (WORDN 3 11))))) in
  let new_R_ccr = ((r_writeB /\
                     (r_reg_sel = (WORDN 3 3))) => BSel(I_ad_in) | R_ccr) in
  let new_R_ccr_rden = (r_readB /\ (r_reg_sel = (WORDN 3 3))) in
  let new_R_gcr = ((r_writeB /\
                     (r_reg_sel = (WORDN 3 2))) => BSel(I_ad_in) | R_gcr) in
  let new_R_gcr_rden = (r_readB /\ (r_reg_sel = (WORDN 3 2))) in

  let new_R_c01_cout_del = R_ctr1_cry in
     let int1_enR = (((ELEMENT new_R_gcr (17)) /\ R_c01_cout_del) \/
                      ~(ELEMENT new_R_gcr (18))) in
     let int1_enS = ((ELEMENT new_R_gcr (18)) /\
                     (R_ctr1_cry /\ (ELEMENT new_R_gcr (16)) \/ r_cir_wr01B)) in
  let new_R_int1_en =
        ((int1_enR \/ int1_enS)
           => ((int1_enS /\ ~int1_enR) => T |
               (~int1_enS /\ int1_enR) => F |
               (~int1_enS /\ ~int1_enR) => F | ARB)
           |  R_int1_en) in
  let new_R_c23_cout_del = R_ctr3_cry in
     let int2_enR = (((ELEMENT new_R_gcr (21)) /\ R_c23_cout_del) \/
                      ~(ELEMENT new_R_gcr (22))) in
     let int2_enS = ((ELEMENT new_R_gcr (22)) /\
                     (R_ctr3_cry /\ (ELEMENT new_R_gcr (20)) \/ r_cir_wr23B)) in
  let new_R_int2_en =
        ((int2_enR \/ int2_enS)
           => ((int2_enS /\ ~int2_enR) => T |
               (~int2_enS /\ int2_enR) => F |
               (~int2_enS /\ ~int2_enR) => F | ARB)
           |  R_int2_en) in
  let new_R_ctr0_in =
        ((r_writeB /\ (r_reg_sel = (WORDN 3 8))) => BSel(I_ad_in) | R_ctr0_in) in
  let new_R_ctr0_mux_sel =
        ((R_ctr1_cry /\ (ELEMENT new_R_gcr (16))) \/ r_cir_wr01B) in
  let new_R_ctr0_irden = (r_readB /\ (r_reg_sel = (WORDN 3 8))) in
  let new_R_ctr0 = ((R_ctr0_mux_sel) => R_ctr0_in | R_ctr0_new) in
  let new_R_ctr0_new =
        (((ELEMENT R_gcr (19))) => (INCN 31 new_R_ctr0) | new_R_ctr0) in
  let new_R_ctr0_cry = ((ELEMENT R_gcr (19)) /\ (ONES 31 new_R_ctr0)) in
  let new_R_ctr0_out = ((r_fsm_cntlatch) => R_ctr0_new | R_ctr0_out) in
  let new_R_ctr0_orden = (r_readB /\ (r_reg_sel = (WORDN 3 12))) in
  let new_R_ctr1_in =
        ((r_writeB /\ (r_reg_sel = (WORDN 3 9))) => BSel(I_ad_in) | R_ctr1_in) in
  let new_R_ctr1_mux_sel =
        ((R_ctr1_cry /\ (ELEMENT new_R_gcr (16))) \/ r_cir_wr01B) in
  let new_R_ctr1_irden = (r_readB /\ (r_reg_sel = (WORDN 3 9))) in
  let new_R_ctr1 = ((R_ctr1_mux_sel) => R_ctr1_in | R_ctr1_new) in
  let new_R_ctr1_new = ((R_ctr0_cry) => (INCN 31 new_R_ctr1) | new_R_ctr1) in
  let new_R_ctr1_cry = (R_ctr0_cry /\ (ONES 31 new_R_ctr1)) in
  let new_R_ctr1_out = ((R_cntlatch_del) => R_ctr1_new | R_ctr1_out) in
  let new_R_ctr1_orden = (r_readB /\ (r_reg_sel = (WORDN 3 13))) in
  let new_R_ctr2_in =
        ((r_writeB /\ (r_reg_sel = (WORDN 3 10))) => BSel(I_ad_in) | R_ctr2_in) in
  let new_R_ctr2_mux_sel =
        ((R_ctr3_cry /\ (ELEMENT new_R_gcr (20))) \/ r_cir_wr23B) in
  let new_R_ctr2_irden = (r_readB /\ (r_reg_sel = (WORDN 3 10))) in
```

```
  let new_R_ctr2 = ((R_ctr2_mux_sel) => R_ctr2_in | R_ctr2_new) in
  let new_R_ctr2_new =
        (((ELEMENT R_gcr (23))) => (INCN 31 new_R_ctr2) | new_R_ctr2) in
  let new_R_ctr2_cry = ((ELEMENT R_gcr (23)) /\ (ONES 31 new_R_ctr2)) in
  let new_R_ctr2_out = ((r_fsm_cntlatch) => R_ctr2_new | R_ctr2_out) in
  let new_R_ctr2_orden = (r_readB /\ (r_reg_sel = (WORDN 3 14))) in
  let new_R_ctr3_in =
        ((r_writeB /\ (r_reg_sel = (WORDN 3 11))) => BSel(I_ad_in) | R_ctr3_in) in
  let new_R_ctr3_mux_sel =
        ((R_ctr3_cry /\ (ELEMENT new_R_gcr (20))) \/ r_cir_wr23B) in
  let new_R_ctr3_irden = (r_readB /\ (r_reg_sel = (WORDN 3 11))) in
  let new_R_ctr3 = ((R_ctr3_mux_sel) => R_ctr3_in | R_ctr3_new) in
  let new_R_ctr3_new = ((R_ctr2_cry) => (INCN 31 new_R_ctr3) | new_R_ctr3) in
  let new_R_ctr3_cry = (R_ctr2_cry /\ (ONES 31 new_R_ctr3)) in
  let new_R_ctr3_out = ((R_cntlatch_del) => R_ctr3_new | R_ctr3_out) in
  let new_R_ctr3_orden = (r_readB /\ (r_reg_sel = (WORDN 3 15))) in
  let new_R_icr_load =
        (r_writeB /\ ((r_reg_sel = (WORDN 3 0)) \/ (r_reg_sel = (WORDN 3 1)))) in
  let new_R_icr_old = ((new_R_icr_load) => R_icr | R_icr_old) in
  let new_R_icr_mask = ((new_R_icr_load) => BSel(I_ad_in) | R_icr_mask) in
  let new_R_icr =
((R_icr_load)
           => ((~(r_reg_sel = (WORDN 3 1))) => (Andn rep (R_icr_old, R_icr_mask))
                                             |  (Orn rep (R_icr_old, R_icr_mask)))
            |  R_icr) in
  let new_R_icr_rden =
        ((new_R_fsm_state = RA) /\
         ((r_reg_sel = (WORDN 3 0)) \/ (r_reg_sel = (WORDN 3 1)))) in
  let sr1_0 = (MALTER ARBN (1,0) (BSel(Cpu_fail))) in
  let sr3_0 = (MALTER sr1_0 (3,2) (BSel(Reset_cpu))) in
  let sr8_0 = (ALTER sr3_0 (8) (BSel(Piu_fail))) in
  let sr9_0 = (ALTER sr8_0 (9) (BSel(Pmm_fail))) in
  let sr15_0 = (MALTER sr9_0 (15,12) (BSel(S_state))) in
  let sr21_0 = (MALTER sr15_0 (21,16) (BSel(Id))) in
  let sr23_0 = (MALTER sr21_0 (23,22) (BSel(ChannelID))) in
  let sr24_0 = (ALTER sr23_0 (24) (BSel(CB_parity))) in
  let sr27_0 = (MALTER sr24_0 (27,25) (BSel(C_ss))) in
  let sr28_0 = (ALTER sr27_0 (28) (BSel(MB_parity))) in
  let new_R_sr = ((r_fsm_cntlatch) => sr28_0 | R_sr) in
  let new_R_sr_rden = (r_readB /\ (r_reg_sel = (WORDN 3 4))) in
  let r_int0_en = (((ELEMENT R_icr (0)) /\ (ELEMENT R_icr (8))) \/
 ((ELEMENT R_icr (1)) /\ (ELEMENT R_icr (9))) \/
 ((ELEMENT R_icr (2)) /\ (ELEMENT R_icr (10))) \/
 ((ELEMENT R_icr (3)) /\ (ELEMENT R_icr (11))) \/
 ((ELEMENT R_icr (4)) /\ (ELEMENT R_icr (12))) \/
 ((ELEMENT R_icr (5)) /\ (ELEMENT R_icr (13))) \/
 ((ELEMENT R_icr (6)) /\ (ELEMENT R_icr (14))) \/
 ((ELEMENT R_icr (7)) /\ (ELEMENT R_icr (15)))) in
  let new_R_int0_dis = r_int0_en in
  let r_int3_en = (((ELEMENT R_icr (16)) /\ (ELEMENT R_icr (24))) \/
 ((ELEMENT R_icr (17)) /\ (ELEMENT R_icr (25))) \/
 ((ELEMENT R_icr (18)) /\ (ELEMENT R_icr (26))) \/
 ((ELEMENT R_icr (19)) /\ (ELEMENT R_icr (27))) \/
 ((ELEMENT R_icr (20)) /\ (ELEMENT R_icr (28))) \/
 ((ELEMENT R_icr (21)) /\ (ELEMENT R_icr (29))) \/
 ((ELEMENT R_icr (22)) /\ (ELEMENT R_icr (30))) \/
 ((ELEMENT R_icr (23)) /\ (ELEMENT R_icr (31)))) in
  let new_R_int3_dis = r_int3_en in

  let new_R_busA_latch =
((R_ctr0_irden) => R_ctr0_in |
  ((R_ctr0_orden) => R_ctr0_out |
  ((R_ctr1_irden) => R_ctr1_in |
  ((R_ctr1_orden) => R_ctr1_out |
  ((R_ctr2_irden) => R_ctr2_in |
  ((R_ctr2_orden) => R_ctr2_out |
  ((R_ctr3_irden) => R_ctr3_in |
  ((R_ctr3_orden) => R_ctr3_out |
  ((R_icr_rden) => R_icr |
  ((R_ccr_rden) => R_ccr |
  ((R_gcr_rden) => R_gcr |
  ((R_sr_rden) => R_sr | ARBN))))))))))))) in
```

```
  let new_R_fsm_ale_ = (BSel(I_rale_)) in
  let new_R_fsm_mrdy_ = (BSel(I_mrdy_)) in
  let new_R_fsm_last_ = (BSel(I_last_)) in
  let new_R_fsm_rst = (BSel(Rst)) in

     (RState new_R_ctr0 new_R_ctr1 new_R_ctr2 new_R_ctr3
             new_R_busA_latch new_R_fsm_state
             new_R_fsm_ale_ new_R_fsm_mrdy_ new_R_fsm_last_ new_R_fsm_rst
             new_R_int0_dis new_R_int3_dis new_R_c01_cout_del new_R_int1_en
             new_R_c23_cout_del new_R_int2_en new_R_wr new_R_cntlatch_del
             new_R_srdy_del_ new_R_reg_sel new_R_ctr0_in new_R_ctr0_mux_sel
             new_R_ctr0_irden new_R_ctr0_cry new_R_ctr0_new new_R_ctr0_out
             new_R_ctr0_orden new_R_ctr1_in new_R_ctr1_mux_sel new_R_ctr1_irden
             new_R_ctr1_cry new_R_ctr1_new new_R_ctr1_out new_R_ctr1_orden
             new_R_ctr2_in new_R_ctr2_mux_sel new_R_ctr2_irden new_R_ctr2_cry
             new_R_ctr2_new new_R_ctr2_out new_R_ctr2_orden new_R_ctr3_in
             new_R_ctr3_mux_sel new_R_ctr3_irden new_R_ctr3_cry new_R_ctr3_new
             new_R_ctr3_out new_R_ctr3_orden new_R_icr_load new_R_icr_old
             new_R_icr_mask new_R_icr new_R_icr_rden new_R_ccr new_R_ccr_rden
             new_R_gcr new_R_gcr_rden new_R_sr new_R_sr_rden)"
   );;

let RClockNSF_REW = save_thm
   ('RClockNSF_REW',
    (PURE_ONCE_REWRITE_RULE [ASel;BSel] RClockNSF)
   );;

%----------------------------------------------------------------------------
Output definition for R-Port instruction.
----------------------------------------------------------------------------%

let RClockOF = new_definition
    ('RClockOF',
  "! (rep :^REP_ty) (s :r_state) (e :r_env) .
      RClockOF rep s e =

      let R_fsm_state = R_fsm_stateS s and
          R_fsm_ale_ = R_fsm_ale_S s and
          R_fsm_mrdy_ = R_fsm_mrdy_S s and
          R_fsm_last_ = R_fsm_last_S s and
          R_fsm_rst = R_fsm_rstS s and
          R_ctr0_in = R_ctr0_inS s and
          R_ctr0_mux_sel = R_ctr0_mux_selS s and
          R_ctr0 = R_ctr0S s and
          R_ctr0_irden = R_ctr0_irdenS s and
          R_ctr0_new = R_ctr0_newS s and
          R_ctr0_cry = R_ctr0_cryS s and
          R_ctr0_out = R_ctr0_outS s and
          R_ctr0_orden = R_ctr0_ordenS s and
          R_ctr1_in = R_ctr1_inS s and
          R_ctr1_mux_sel = R_ctr1_mux_selS s and
          R_ctr1 = R_ctr1S s and
          R_ctr1_irden = R_ctr1_irdenS s and
          R_ctr1_new = R_ctr1_newS s and
          R_ctr1_cry = R_ctr1_cryS s and
          R_ctr1_out = R_ctr1_outS s and
          R_ctr1_orden = R_ctr1_ordenS s and
          R_ctr2_in = R_ctr2_inS s and
          R_ctr2_mux_sel = R_ctr2_mux_selS s and
          R_ctr2 = R_ctr2S s and
          R_ctr2_irden = R_ctr2_irdenS s and
          R_ctr2_new = R_ctr2_newS s and
          R_ctr2_cry = R_ctr2_cryS s and
          R_ctr2_out = R_ctr2_outS s and
          R_ctr2_orden = R_ctr2_ordenS s and
          R_ctr3_in = R_ctr3_inS s and
          R_ctr3_mux_sel = R_ctr3_mux_selS s and
          R_ctr3 = R_ctr3S s and
          R_ctr3_irden = R_ctr3_irdenS s and
          R_ctr3_new = R_ctr3_newS s and
          R_ctr3_cry = R_ctr3_cryS s and
          R_ctr3_out = R_ctr3_outS s and
```

```
        R_ctr3_orden = R_ctr3_ordenS s and
        R_icr_load = R_icr_loadS s and
        R_icr_old = R_icr_oldS s and
        R_icr_mask = R_icr_maskS s and
        R_icr_rden = R_icr_rdenS s and
        R_icr = R_icrS s and
        R_ccr = R_ccrS s and
        R_ccr_rden = R_ccr_rdenS s and
        R_gcr = R_gcrS s and
        R_gcr_rden = R_gcr_rdenS s and
        R_sr = R_srS s and
        R_sr_rden = R_sr_rdenS s and
        R_int0_dis = R_int0_disS s and
        R_int3_dis = R_int3_disS s and
        R_c01_cout_del = R_c01_cout_delS s and
        R_int1_en = R_int1_enS s and
        R_c23_cout_del = R_c23_cout_delS s and
        R_int2_en = R_int2_enS s and
        R_wr = R_wrS s and
        R_cntlatch_del = R_cntlatch_delS s and
        R_srdy_del_ = R_srdy_del_S s and
        R_reg_sel = R_reg_selS s and
        R_busA_latch = R_busA_latchS s in
    let Rst = RstE e and
        I_ad_in = I_ad_inE e and
        I_rale_ = I_rale_E e and
        I_last_ = I_last_E e and
        I_be_ = I_be_E e and
        I_mrdy_ = I_mrdy_E e and
        Disable_int = Disable_intE e and
        Disable_writes = Disable_writesE e and
        Cpu_fail = Cpu_failE e and
        Reset_cpu = Reset_cpuE e and
        Piu_fail = Piu_failE e and
        Pmm_fail = Pmm_failE e and
        S_state = S_stateE e and
        Id = IdE e and
        ChannelID = ChannelIDE e and
        CB_parity = CB_parityE e and
        MB_parity = MB_parityE e and
        C_ss = C_ssE e in

  let new_R_fsm_state =
((R_fsm_rst) => RI |
  ((R_fsm_state = RI) => ((~R_fsm_ale_) => RA | RI) |
  ((R_fsm_state = RA) => ((~R_fsm_mrdy_) => RD | RA) |
  ((~R_fsm_last_) => RI | RA)))) in
  let r_fsm_cntlatch = ((R_fsm_state = RI) /\ ~R_fsm_ale_) in
  let r_fsm_srdy_ = ~((R_fsm_state = RA) /\ ~R_fsm_mrdy_) in
  let new_R_wr = ((~BSel(I_rale_)) => (ELEMENT (BSel(I_ad_in)) (27)) | R_wr)
      in
  let new_R_cntlatch_del = r_fsm_cntlatch in
  let new_R_srdy_del_ = r_fsm_srdy_ in
  let new_R_reg_sel =
((~BSel(I_rale_)) => (SUBARRAY (BSel(I_ad_in)) (3,0)) |
 ((~R_srdy_del_) => (INCN 3 R_reg_sel) | R_reg_sel)) in
  let r_reg_sel = ((~R_srdy_del_) => (INCN 3 R_reg_sel) | R_reg_sel) in
  let r_writeA = (~ASel(Disable_writes) /\ R_wr /\ (new_R_fsm_state = RD)) in
  let r_writeB = (~BSel(Disable_writes) /\ new_R_wr /\
                  (new_R_fsm_state = RD)) in
  let r_readA = (~R_wr /\ (new_R_fsm_state = RA)) in
  let r_readB = (~new_R_wr /\ (new_R_fsm_state = RA)) in

  let r_cir_wr01A = ((r_writeA /\ ((r_reg_sel = (WORDN 3 8)) \/
                                   (r_reg_sel = (WORDN 3 9))))) in
  let r_cir_wr01B = ((r_writeB /\ ((r_reg_sel = (WORDN 3 8)) \/
                                   (r_reg_sel = (WORDN 3 9))))) in
  let r_cir_wr23A = ((r_writeA /\ ((r_reg_sel = (WORDN 3 10)) \/
                                   (r_reg_sel = (WORDN 3 11))))) in
  let r_cir_wr23B = ((r_writeB /\ ((r_reg_sel = (WORDN 3 10)) \/
                                   (r_reg_sel = (WORDN 3 11))))) in
  let new_R_ccr = ((r_writeB /\
```

```
                    (r_reg_sel = (WORDN 3 3))) => BSel(I_ad_in) | R_ccr) in
let new_R_ccr_rden = (r_readB /\ (r_reg_sel = (WORDN 3 3))) in
let new_R_gcr = ((r_writeB /\
                    (r_reg_sel = (WORDN 3 2))) => BSel(I_ad_in) | R_gcr) in
let new_R_gcr_rden = (r_readB /\ (r_reg_sel = (WORDN 3 2))) in

let new_R_c01_cout_del = R_ctr1_cry in
   let int1_enR = (((ELEMENT new_R_gcr (17)) /\ R_c01_cout_del) \/
                    ~(ELEMENT new_R_gcr (18))) in
   let int1_enS = ((ELEMENT new_R_gcr (18)) /\
                    (R_ctr1_cry /\ (ELEMENT new_R_gcr (16)) \/ r_cir_wr01B)) in
let new_R_int1_en =
      ((int1_enR \/ int1_enS)
         => ((int1_enS /\ ~int1_enR) => T |
             (~int1_enS /\ int1_enR) => F |
             (~int1_enS /\ ~int1_enR) => F | ARB)
         |  R_int1_en) in
let new_R_c23_cout_del = R_ctr3_cry in
   let int2_enR = (((ELEMENT new_R_gcr (21)) /\ R_c23_cout_del) \/
                    ~(ELEMENT new_R_gcr (22))) in
   let int2_enS = ((ELEMENT new_R_gcr (22)) /\
                    (R_ctr3_cry /\ (ELEMENT new_R_gcr (20)) \/ r_cir_wr23B)) in
let new_R_int2_en =
      ((int2_enR \/ int2_enS)
         => ((int2_enS /\ ~int2_enR) => T |
             (~int2_enS /\ int2_enR) => F |
             (~int2_enS /\ ~int2_enR) => F | ARB)
         |  R_int2_en) in
let new_R_ctr0_in =
      ((r_writeB /\ (r_reg_sel = (WORDN 3 8))) => BSel(I_ad_in) | R_ctr0_in) in
let new_R_ctr0_mux_sel =
      ((R_ctr1_cry /\ (ELEMENT new_R_gcr (16))) \/ r_cir_wr01B) in
let new_R_ctr0_irden = (r_readB /\ (r_reg_sel = (WORDN 3 8))) in
let new_R_ctr0 = ((R_ctr0_mux_sel) => R_ctr0_in | R_ctr0_new) in
let new_R_ctr0_new =
      (((ELEMENT R_gcr (19))) => (INCN 31 new_R_ctr0) | new_R_ctr0) in
let new_R_ctr0_cry = ((ELEMENT R_gcr (19)) /\ (ONES 31 new_R_ctr0)) in
let new_R_ctr0_out = ((r_fsm_cntlatch) => R_ctr0_new | R_ctr0_out) in
let new_R_ctr0_orden = (r_readB /\ (r_reg_sel = (WORDN 3 12))) in
let new_R_ctr1_in =
      ((r_writeB /\ (r_reg_sel = (WORDN 3 9))) => BSel(I_ad_in) | R_ctr1_in) in
let new_R_ctr1_mux_sel =
      ((R_ctr1_cry /\ (ELEMENT new_R_gcr (16))) \/ r_cir_wr01B) in
let new_R_ctr1_irden = (r_readB /\ (r_reg_sel = (WORDN 3 9))) in
let new_R_ctr1 = ((R_ctr1_mux_sel) => R_ctr1_in | R_ctr1_new) in
let new_R_ctr1_new = ((R_ctr0_cry) => (INCN 31 new_R_ctr1) | new_R_ctr1) in
let new_R_ctr1_cry = (R_ctr0_cry /\ (ONES 31 new_R_ctr1)) in
let new_R_ctr1_out = ((R_cntlatch_del) => R_ctr1_new | R_ctr1_out) in
let new_R_ctr1_orden = (r_readB /\ (r_reg_sel = (WORDN 3 13))) in
let new_R_ctr2_in =
      ((r_writeB /\ (r_reg_sel = (WORDN 3 10))) => BSel(I_ad_in) | R_ctr2_in) in
let new_R_ctr2_mux_sel =
      ((R_ctr3_cry /\ (ELEMENT new_R_gcr (20))) \/ r_cir_wr23B) in
let new_R_ctr2_irden = (r_readB /\ (r_reg_sel = (WORDN 3 10))) in
let new_R_ctr2 = ((R_ctr2_mux_sel) => R_ctr2_in | R_ctr2_new) in
let new_R_ctr2_new =
      (((ELEMENT R_gcr (23))) => (INCN 31 new_R_ctr2) | new_R_ctr2) in
let new_R_ctr2_cry = ((ELEMENT R_gcr (23)) /\ (ONES 31 new_R_ctr2)) in
let new_R_ctr2_out = ((r_fsm_cntlatch) => R_ctr2_new | R_ctr2_out) in
let new_R_ctr2_orden = (r_readB /\ (r_reg_sel = (WORDN 3 14))) in
let new_R_ctr3_in =
      ((r_writeB /\ (r_reg_sel = (WORDN 3 11))) => BSel(I_ad_in) | R_ctr3_in) in
let new_R_ctr3_mux_sel =
      ((R_ctr3_cry /\ (ELEMENT new_R_gcr (20))) \/ r_cir_wr23B) in
let new_R_ctr3_irden = (r_readB /\ (r_reg_sel = (WORDN 3 11))) in
let new_R_ctr3 = ((R_ctr3_mux_sel) => R_ctr3_in | R_ctr3_new) in
let new_R_ctr3_new = ((R_ctr2_cry) => (INCN 31 new_R_ctr3) | new_R_ctr3) in
let new_R_ctr3_cry = (R_ctr2_cry /\ (ONES 31 new_R_ctr3)) in
let new_R_ctr3_out = ((R_cntlatch_del) => R_ctr3_new | R_ctr3_out) in
let new_R_ctr3_orden = (r_readB /\ (r_reg_sel = (WORDN 3 15))) in
let new_R_icr_load =
      (r_writeB /\ ((r_reg_sel = (WORDN 3 0)) \/ (r_reg_sel = (WORDN 3 1)))) in
```

```
  let new_R_icr_old = ((new_R_icr_load) => R_icr | R_icr_old) in
  let new_R_icr_mask = ((new_R_icr_load) => BSel(I_ad_in) | R_icr_mask) in
  let new_R_icr =
((R_icr_load)
           => ((~(r_reg_sel = (WORDN 3 1))) => (Andn rep (R_icr_old, R_icr_mask))
                                             |  (Orn rep (R_icr_old, R_icr_mask)))
           |  R_icr) in
  let new_R_icr_rden =
       ((new_R_fsm_state = RA) /\
        ((r_reg_sel = (WORDN 3 0)) \/ (r_reg_sel = (WORDN 3 1)))) in
  let sr1_0 = (MALTER ARBN (1,0) (BSel(Cpu_fail))) in
  let sr3_0 = (MALTER sr1_0 (3,2) (BSel(Reset_cpu))) in
  let sr8_0 = (ALTER sr3_0 (8) (BSel(Piu_fail))) in
  let sr9_0 = (ALTER sr8_0 (9) (BSel(Pmm_fail))) in
  let sr15_0 = (MALTER sr9_0 (15,12) (BSel(S_state))) in
  let sr21_0 = (MALTER sr15_0 (21,16) (BSel(Id))) in
  let sr23_0 = (MALTER sr21_0 (23,22) (BSel(ChannelID))) in
  let sr24_0 = (ALTER sr23_0 (24) (BSel(CB_parity))) in
  let sr27_0 = (MALTER sr24_0 (27,25) (BSel(C_ss))) in
  let sr28_0 = (ALTER sr27_0 (28) (BSel(MB_parity))) in
  let new_R_sr = ((r_fsm_cntlatch) => sr28_0 | R_sr) in
  let new_R_sr_rden = (r_readB /\ (r_reg_sel = (WORDN 3 4))) in
  let r_int0_en = (((ELEMENT R_icr (0)) /\ (ELEMENT R_icr (8))) \/
 ((ELEMENT R_icr (1)) /\ (ELEMENT R_icr (9))) \/
 ((ELEMENT R_icr (2)) /\ (ELEMENT R_icr (10))) \/
 ((ELEMENT R_icr (3)) /\ (ELEMENT R_icr (11))) \/
 ((ELEMENT R_icr (4)) /\ (ELEMENT R_icr (12))) \/
 ((ELEMENT R_icr (5)) /\ (ELEMENT R_icr (13))) \/
 ((ELEMENT R_icr (6)) /\ (ELEMENT R_icr (14))) \/
 ((ELEMENT R_icr (7)) /\ (ELEMENT R_icr (15)))) in
  let new_R_int0_dis = r_int0_en in
  let r_int3_en = (((ELEMENT R_icr (16)) /\ (ELEMENT R_icr (24))) \/
 ((ELEMENT R_icr (17)) /\ (ELEMENT R_icr (25))) \/
 ((ELEMENT R_icr (18)) /\ (ELEMENT R_icr (26))) \/
 ((ELEMENT R_icr (19)) /\ (ELEMENT R_icr (27))) \/
 ((ELEMENT R_icr (20)) /\ (ELEMENT R_icr (28))) \/
 ((ELEMENT R_icr (21)) /\ (ELEMENT R_icr (29))) \/
 ((ELEMENT R_icr (22)) /\ (ELEMENT R_icr (30))) \/
 ((ELEMENT R_icr (23)) /\ (ELEMENT R_icr (31)))) in
  let new_R_int3_dis = r_int3_en in

  let new_R_busA_latch =
((R_ctr0_irden) => R_ctr0_in |
  ((R_ctr0_orden) => R_ctr0_out |
  ((R_ctr1_irden) => R_ctr1_in |
  ((R_ctr1_orden) => R_ctr1_out |
  ((R_ctr2_irden) => R_ctr2_in |
  ((R_ctr2_orden) => R_ctr2_out |
  ((R_ctr3_irden) => R_ctr3_in |
  ((R_ctr3_orden) => R_ctr3_out |
  ((R_icr_rden) => R_icr |
  ((R_ccr_rden) => R_ccr |
  ((R_gcr_rden) => R_gcr |
  ((R_sr_rden) => R_sr | ARBN))))))))))))) in
  let new_R_fsm_ale_ = (BSel(I_rale_)) in
  let new_R_fsm_mrdy_ = (BSel(I_mrdy_)) in
  let new_R_fsm_last_ = (BSel(I_last_)) in
  let new_R_fsm_rst = (BSel(Rst)) in

     let I_ad_out =
        (((~R_wr /\ ((new_R_fsm_state = RA) \/ (new_R_fsm_state = RD)))
           => (BUSN new_R_busA_latch) | Offn),
         ((~new_R_wr /\ ((new_R_fsm_state = RA) \/ (new_R_fsm_state = RD)))
           => (BUSN new_R_busA_latch) | Offn)) in
     let I_srdy_ =
        ((((new_R_fsm_state = RA) \/ (new_R_fsm_state = RD))
           => (WIRE ~((R_fsm_state = RA) /\ ~R_fsm_mrdy_)) | Z),
         (((new_R_fsm_state = RA) \/ (new_R_fsm_state = RD))
           => (WIRE ~((R_fsm_state = RA) /\ ~R_fsm_mrdy_)) | Z)) in
     let Int0_ =
        ((~(r_int0_en /\ ~R_int0_dis /\ ~ASel(Disable_int))),
         (~(r_int0_en /\ ~R_int0_dis /\ ~BSel(Disable_int)))) in
```

```
     let Int1 =
        ((R_ctr1_cry /\ R_int1_en /\ ~ASel(Disable_int)),
         (R_ctr1_cry /\ new_R_int1_en /\ ~BSel(Disable_int))) in
     let Int2 =
        ((R_ctr3_cry /\ R_int2_en /\ ~ASel(Disable_int)),
         (R_ctr3_cry /\ new_R_int2_en /\ ~BSel(Disable_int))) in
     let Int3_ =
        ((~(r_int3_en /\ ~R_int3_dis /\ ~ASel(Disable_int))),
         (~(r_int3_en /\ ~R_int3_dis /\ ~BSel(Disable_int)))) in
     let Ccr = (R_ccr, new_R_ccr) in
     let Led = ((SUBARRAY R_gcr (3,0)), (SUBARRAY new_R_gcr (3,0))) in
     let Reset_error = ((ELEMENT R_gcr (24)), (ELEMENT new_R_gcr (24))) in
     let Pmm_invalid = ((ELEMENT R_gcr (28)), (ELEMENT new_R_gcr (28))) in

  (ROut I_ad_out I_srdy_ Int0_ Int1 Int2 Int3_ Ccr Led Reset_error
        Pmm_invalid)"
   );;

let RClockOF_REW = save_thm
   ('RClockOF_REW',
    (PURE_ONCE_REWRITE_RULE [ASel;BSel] RClockOF)
   );;

let RC_Exec = new_definition
   ('RC_Exec',
    "! (rci :RCI) (s :timeC->r_state) (e :timeC->r_env) (p :timeC->r_out)
       (t :timeC) .
     RC_Exec rci s e p t = T"
   );;

let RC_PreC = new_definition
   ('RC_PreC',
    "! (rci :RCI) (s :timeC->r_state) (e :timeC->r_env) (p :timeC->r_out)
       (t :timeC) .
     RC_PreC rci s e p t = T"
   );;

let RC_PostC = new_definition
   ('RC_PostC',
    "! (rci :RCI) (s :timeC->r_state) (e :timeC->r_env) (p :timeC->r_out)
       (t :timeC) .
     RC_PostC rci s e p t =
        (s (t+1) = RClockNSF (s t) (e t)) /\
        (p t = RClockOF (s t) (e t))"
   );;
 let RC_Correct = new_definition
   ('RC_Correct',
    "! (rci :RCI) (s :timeC->r_state) (e :timeC->r_env) (p :timeC->r_out)
       (t :timeC) .
     RC_Correct rci s e p t =
        RC_Exec rci s e p t /\
        RC_PreC rci s e p t
        ==>
        RC_PostC rci s e p t"
   );;

let RCSet_Correct = new_definition
   ('RCSet_Correct',
    "! (s :timeC->r_state) (e :timeC->r_env) (p :timeC->r_out) .
     RCSet_Correct s e p = !(rci:RCI)(t:timeC). RC_Correct rci s e p t"
   );;


close_theory();;
```

3.5 C-Port Definitions

This section contains the theories *caux_def*, *cblock_def*, and *cclock_def*, defining the C-Port design.

```
%-----------------------------------------------------------------------------

   File:        caux_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        3 March 1993

   This file contains auxiliary definitions for the C-Port of the FTEP PIU, an
   ASIC developed by the Embedded Processing Laboratory, Boeing High Technology
   Center.

-----------------------------------------------------------------------------%

set_flag ('timing', true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm caux_def.th';;

new_theory 'caux_def';;

map new_parent ['array_def';'wordn_def';'busn_def';'ineq'];;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('wordn', ":(num->bool)");;
new_type_abbrev ('busn', ":(num->wire)");;

let MSTART = "WORDN 2 4";;
let MEND = "WORDN 2 5";;
let MRDY = "WORDN 2 6";;
let MWAIT = "WORDN 2 7";;
let MABORT = "WORDN 2 0";;

let SACK = "WORDN 2 5";;
let SRDY = "WORDN 2 6";;
let SWAIT = "WORDN 2 7";;
let SABORT = "WORDN 2 0";;

%-----------------------------------------------------------------------------
   Abstract data types for the C-Port Master, Slave, and SRdyEn FSMs.
-----------------------------------------------------------------------------%

let cmfsm_ty_Axiom =
   define_type 'cmfsm_ty_Axiom'
               'cmfsm_ty = CMI | CMR | CMA3 | CMA1 | CMA0 | CMA2 | CMD1 | CMD0 |
                           CMW | CMABT';;

let csfsm_ty_Axiom =
   define_type 'csfsm_ty_Axiom'
               'csfsm_ty = CSI | CSL | CSA1 | CSA0 | CSA0W | CSALE | CSRR |
                           CSD1 | CSD0 | CSACK | CSABT';;

let cefsm_ty_Axiom =
   define_type 'cefsm_ty_Axiom'
               'cefsm_ty = CEI | CEE';;

%-----------------------------------------------------------------------------
   Abstract data type for the C-Port instruction.
-----------------------------------------------------------------------------%
let CCI =
   define_type 'CCI'
               'CCI = CC_X';;
```

```
%-----------------------------------------------------------------------------
   Abstract data type for the state.
-----------------------------------------------------------------------------%

let cc_state =
   define_type 'cc_state'
               'cc_state = CCState cmfsm_ty bool bool bool bool bool bool bool bool
                                  bool bool wordn bool csfsm_ty bool bool bool
                                  bool bool bool wordn csfsm_ty bool bool bool
                                  bool bool bool bool bool wordn bool bool bool
                                  bool bool bool bool bool bool bool bool bool
                                  wordn wordn wordn wordn wordn wordn wordn
                                  bool';;

let C_mfsm_stateS = new_recursive_definition
    false
    cc_state
    'C_mfsm_stateS'
    "C_mfsm_stateS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_state";;

let C_mfsm_srdy_enS = new_recursive_definition
    false
    cc_state
    'C_mfsm_srdy_enS'
    "C_mfsm_srdy_enS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_srdy_en";;

let C_mfsm_DS = new_recursive_definition
    false
    cc_state
    'C_mfsm_DS'
    "C_mfsm_DS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_D";;

let C_mfsm_grantS = new_recursive_definition
    false
    cc_state
    'C_mfsm_grantS'
```

```
    "C_mfsm_grantS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_grant";;

let C_mfsm_rstS = new_recursive_definition
    false
    cc_state
    'C_mfsm_rstS'
    "C_mfsm_rstS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_rst";;

let C_mfsm_busyS = new_recursive_definition
    false
    cc_state
    'C_mfsm_busyS'
    "C_mfsm_busyS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_busy";;

let C_mfsm_writeS = new_recursive_definition
    false
    cc_state
    'C_mfsm_writeS'
    "C_mfsm_writeS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_write";;

let C_mfsm_crqt_S = new_recursive_definition
    false
    cc_state
    'C_mfsm_crqt_S'
```

```
    "C_mfsm_crqt_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_crqt_";;

let C_mfsm_hold_S = new_recursive_definition
    false
    cc_state
    'C_mfsm_hold_S'
    "C_mfsm_hold_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_hold_";;

let C_mfsm_last_S = new_recursive_definition
    false
    cc_state
    'C_mfsm_last_S'
    "C_mfsm_last_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_last_";;

let C_mfsm_lock_S = new_recursive_definition
    false
    cc_state
    'C_mfsm_lock_S'
    "C_mfsm_lock_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_lock_";;

let C_mfsm_ssS = new_recursive_definition
    false
    cc_state
    'C_mfsm_ssS'
```

```
    "C_mfsm_ssS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                         C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                         C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                         C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                         C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                         C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                         C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                         C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                         C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                         C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                         C_parity C_source C_data_in C_sizewrbe C_iad_out
                         C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_ss";;

let C_mfsm_invalidS = new_recursive_definition
    false
    cc_state
    'C_mfsm_invalidS'
    "C_mfsm_invalidS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                         C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                         C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                         C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                         C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                         C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                         C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                         C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                         C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                         C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                         C_parity C_source C_data_in C_sizewrbe C_iad_out
                         C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mfsm_invalid";;

let C_sfsm_stateS = new_recursive_definition
    false
    cc_state
    'C_sfsm_stateS'
    "C_sfsm_stateS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                         C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                         C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                         C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                         C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                         C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                         C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                         C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                         C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                         C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                         C_parity C_source C_data_in C_sizewrbe C_iad_out
                         C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_state";;

let C_sfsm_DS = new_recursive_definition
    false
    cc_state
    'C_sfsm_DS'
    "C_sfsm_DS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                         C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                         C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                         C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                         C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                         C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                         C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                         C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                         C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                         C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                         C_parity C_source C_data_in C_sizewrbe C_iad_out
                         C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_D";;

let C_sfsm_grantS = new_recursive_definition
    false
    cc_state
    'C_sfsm_grantS'
```

```
    "C_sfsm_grantS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_grant";;

let C_sfsm_rstS = new_recursive_definition
    false
    cc_state
    'C_sfsm_rstS'
    "C_sfsm_rstS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_rst";;

let C_sfsm_writeS = new_recursive_definition
    false
    cc_state
    'C_sfsm_writeS'
    "C_sfsm_writeS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_write";;

let C_sfsm_addressedS = new_recursive_definition
    false
    cc_state
    'C_sfsm_addressedS'
    "C_sfsm_addressedS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                            C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                            C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                            C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                            C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                            C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                            C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                            C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                            C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                            C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                            C_parity C_source C_data_in C_sizewrbe C_iad_out
                            C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_addressed";;

let C_sfsm_hlda_S = new_recursive_definition
    false
    cc_state
    'C_sfsm_hlda_S'
```

```
    "C_sfsm_hlda_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_hlda_";;

let C_sfsm_msS = new_recursive_definition
    false
    cc_state
    'C_sfsm_msS'
    "C_sfsm_msS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sfsm_ms";;

let C_efsm_stateS = new_recursive_definition
    false
    cc_state
    'C_efsm_stateS'
    "C_efsm_stateS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_efsm_state";;

let C_efsm_cale_S = new_recursive_definition
    false
    cc_state
    'C_efsm_cale_S'
    "C_efsm_cale_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_efsm_cale_";;

let C_efsm_last_S = new_recursive_definition
    false
    cc_state
    'C_efsm_last_S'
```

```
    "C_efsm_last_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_efsm_last_";;

let C_efsm_male_S = new_recursive_definition
    false
    cc_state
    'C_efsm_male_S'
    "C_efsm_male_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_efsm_male_";;

let C_efsm_rale_S = new_recursive_definition
    false
    cc_state
    'C_efsm_rale_S'
    "C_efsm_rale_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_efsm_rale_";;

let C_efsm_srdy_S = new_recursive_definition
    false
    cc_state
    'C_efsm_srdy_S'
    "C_efsm_srdy_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_efsm_srdy_";;

let C_efsm_rstS = new_recursive_definition
    false
    cc_state
    'C_efsm_rstS'
```

```
    "C_efsm_rstS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_efsm_rst";;

let C_lock_in_S = new_recursive_definition
    false
    cc_state
    'C_lock_in_S'
    "C_lock_in_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_lock_in_";;

let C_last_in_S = new_recursive_definition
    false
    cc_state
    'C_last_in_S'
    "C_last_in_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_last_in_";;

let C_ssS = new_recursive_definition
    false
    cc_state
    'C_ssS'
    "C_ssS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_ss";;

let C_clkAS = new_recursive_definition
    false
    cc_state
    'C_clkAS'
```

```
    "C_clkAS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_clkA";;

let C_last_out_S = new_recursive_definition
    false
    cc_state
    'C_last_out_S'
    "C_last_out_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_last_out_";;

let C_sidle_delS = new_recursive_definition
    false
    cc_state
    'C_sidle_delS'
    "C_sidle_delS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sidle_del";;

let C_mrqt_delS = new_recursive_definition
    false
    cc_state
    'C_mrqt_delS'
    "C_mrqt_delS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_mrqt_del";;

let C_hold_S = new_recursive_definition
    false
    cc_state
    'C_hold_S'
```

```
    "C_hold_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_hold_";;

let C_cout_0_le_delS = new_recursive_definition
    false
    cc_state
    'C_cout_0_le_delS'
    "C_cout_0_le_delS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_cout_0_le_del";;

let C_cin_2_leS = new_recursive_definition
    false
    cc_state
    'C_cin_2_leS'
    "C_cin_2_leS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_cin_2_le";;

let C_mrdy_del_S = new_recursive_definition
    false
    cc_state
    'C_mrdy_del_S'
    "C_mrdy_del_S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                          C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                          C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                          C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                          C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                          C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                          C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                          C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                          C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                          C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                          C_parity C_source C_data_in C_sizewrbe C_iad_out
                          C_a1a0 C_a3a2 C_iad_in C_wr)
      = C_mrdy_del_";;

let C_iad_en_s_delS = new_recursive_definition
    false
    cc_state
    'C_iad_en_s_delS'
```

```
    "C_iad_en_s_delS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_iad_en_s_del";;

let C_wrdyS = new_recursive_definition
    false
    cc_state
    'C_wrdyS'
    "C_wrdyS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_wrdy";;

let C_rrdyS = new_recursive_definition
    false
    cc_state
    'C_rrdyS'
    "C_rrdyS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_rrdy";;

let C_parityS = new_recursive_definition
    false
    cc_state
    'C_parityS'
    "C_parityS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_parity";;

let C_sourceS = new_recursive_definition
    false
    cc_state
    'C_sourceS'
```

```
    "C_sourceS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                       C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                       C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                       C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                       C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                       C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                       C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                       C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                       C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                       C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                       C_parity C_source C_data_in C_sizewrbe C_iad_out
                       C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_source";;

let C_data_inS = new_recursive_definition
    false
    cc_state
    'C_data_inS'
    "C_data_inS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                       C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                       C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                       C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                       C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                       C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                       C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                       C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                       C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                       C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                       C_parity C_source C_data_in C_sizewrbe C_iad_out
                       C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_data_in";;

let C_sizewrbeS = new_recursive_definition
    false
    cc_state
    'C_sizewrbeS'
    "C_sizewrbeS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                       C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                       C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                       C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                       C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                       C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                       C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                       C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                       C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                       C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                       C_parity C_source C_data_in C_sizewrbe C_iad_out
                       C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_sizewrbe";;

let C_iad_outS = new_recursive_definition
    false
    cc_state
    'C_iad_outS'
    "C_iad_outS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                       C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                       C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                       C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                       C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                       C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                       C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                       C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                       C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                       C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                       C_parity C_source C_data_in C_sizewrbe C_iad_out
                       C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_iad_out";;

let C_a1a0S = new_recursive_definition
    false
    cc_state
    'C_a1a0S'
```

```
    "C_a1a0S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_a1a0";;

let C_a3a2S = new_recursive_definition
    false
    cc_state
    'C_a3a2S'
    "C_a3a2S (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_a3a2";;

let C_iad_inS = new_recursive_definition
    false
    cc_state
    'C_iad_inS'
    "C_iad_inS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_iad_in";;

let C_wrS = new_recursive_definition
    false
    cc_state
    'C_wrS'
    "C_wrS (CCState C_mfsm_state C_mfsm_srdy_en C_mfsm_D C_mfsm_grant
                           C_mfsm_rst C_mfsm_busy C_mfsm_write C_mfsm_crqt_
                           C_mfsm_hold_ C_mfsm_last_ C_mfsm_lock_ C_mfsm_ss
                           C_mfsm_invalid C_sfsm_state C_sfsm_D C_sfsm_grant
                           C_sfsm_rst C_sfsm_write C_sfsm_addressed C_sfsm_hlda_
                           C_sfsm_ms C_efsm_state C_efsm_cale_ C_efsm_last_
                           C_efsm_male_ C_efsm_rale_ C_efsm_srdy_ C_efsm_rst
                           C_lock_in_ C_last_in_ C_ss C_clkA C_last_out_
                           C_sidle_del C_mrqt_del C_hold_ C_cout_0_le_del
                           C_cin_2_le C_mrdy_del_ C_iad_en_s_del C_wrdy C_rrdy
                           C_parity C_source C_data_in C_sizewrbe C_iad_out
                           C_a1a0 C_a3a2 C_iad_in C_wr)
     = C_wr";;

let State_CASES =
   prove_cases_thm (prove_induction_thm cc_state);;

let CCState_Selectors_Work = prove_thm
```

```
   ('CCState_Selectors_Work',
    "! s:cc_state .
     s = (CCState (C_mfsm_stateS s) (C_mfsm_srdy_enS s) (C_mfsm_DS s)
                  (C_mfsm_grantS s) (C_mfsm_rstS s) (C_mfsm_busyS s)
                  (C_mfsm_writeS s) (C_mfsm_crqt_S s) (C_mfsm_hold_S s)
                  (C_mfsm_last_S s) (C_mfsm_lock_S s) (C_mfsm_ssS s)
                  (C_mfsm_invalidS s) (C_sfsm_stateS s) (C_sfsm_DS s)
                  (C_sfsm_grantS s) (C_sfsm_rstS s) (C_sfsm_writeS s)
                  (C_sfsm_addressedS s) (C_sfsm_hlda_S s) (C_sfsm_msS s)
                  (C_efsm_stateS s) (C_efsm_cale_S s) (C_efsm_last_S s)
                  (C_efsm_male_S s) (C_efsm_rale_S s) (C_efsm_srdy_S s)
                  (C_efsm_rstS s) (C_lock_in_S s) (C_last_in_S s) (C_ssS s)
                  (C_clkAS s) (C_last_out_S s) (C_sidle_delS s) (C_mrqt_delS s)
                  (C_hold_S s) (C_cout_0_le_delS s) (C_cin_2_leS s)
                  (C_mrdy_del_S s) (C_iad_en_s_delS s) (C_wrdyS s) (C_rrdyS s)
                  (C_parityS s) (C_sourceS s) (C_data_inS s) (C_sizewrbeS s)
                  (C_iad_outS s) (C_a1a0S s) (C_a3a2S s) (C_iad_inS s)
                  (C_wrS s))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "s:cc_state" State_CASES)
     THEN REWRITE_TAC [C_mfsm_stateS; C_mfsm_srdy_enS; C_mfsm_DS; C_mfsm_grantS;
                       C_mfsm_rstS; C_mfsm_busyS; C_mfsm_writeS; C_mfsm_crqt_S;
                       C_mfsm_hold_S; C_mfsm_last_S; C_mfsm_lock_S; C_mfsm_ssS;
                       C_mfsm_invalidS; C_sfsm_stateS; C_sfsm_DS; C_sfsm_grantS;
                       C_sfsm_rstS; C_sfsm_writeS; C_sfsm_addressedS;
                       C_sfsm_hlda_S; C_sfsm_msS; C_efsm_stateS; C_efsm_cale_S;
                       C_efsm_last_S; C_efsm_male_S; C_efsm_rale_S;
                       C_efsm_srdy_S; C_efsm_rstS; C_lock_in_S; C_last_in_S;
                       C_ssS; C_clkAS; C_last_out_S; C_sidle_delS; C_mrqt_delS;
                       C_hold_S; C_cout_0_le_delS; C_cin_2_leS; C_mrdy_del_S;
                       C_iad_en_s_delS; C_wrdyS; C_rrdyS; C_parityS; C_sourceS;
                       C_data_inS; C_sizewrbeS; C_iad_outS; C_a1a0S; C_a3a2S;
                       C_iad_inS; C_wrS]
   );;

%-----------------------------------------------------------------------------
   Abstract data type for the environment.
-----------------------------------------------------------------------------%

let cc_env =
   define_type 'cc_env'
               'cc_env = CCEnv wordn#wordn wordn#wordn bool#bool bool#bool
                               bool#bool bool#bool bool#bool bool#bool bool#bool
                               bool#bool bool#bool wordn#wordn wordn#wordn
                               wordn#wordn wordn#wordn bool#bool bool#bool
                               wordn#wordn wordn#wordn bool#bool bool#bool
                               wordn#wordn bool#bool';;

let I_ad_inE = new_recursive_definition
    false
    cc_env
    'I_ad_inE'
    "I_ad_inE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                     I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                     CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                     Pmm_failure Piu_invalid Ccr Reset_error)
     = I_ad_in";;

let I_be_in_E = new_recursive_definition
    false
    cc_env
    'I_be_in_E'
    "I_be_in_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                      I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                      CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                      Pmm_failure Piu_invalid Ccr Reset_error)
     = I_be_in_";;

let I_mrdy_in_E = new_recursive_definition
    false
    cc_env
    'I_mrdy_in_E'
```

```
    "I_mrdy_in_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = I_mrdy_in_";;

let I_rale_in_E = new_recursive_definition
    false
    cc_env
    'I_rale_in_E'
    "I_rale_in_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = I_rale_in_";;

let I_male_in_E = new_recursive_definition
    false
    cc_env
    'I_male_in_E'
    "I_male_in_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = I_male_in_";;

let I_last_in_E = new_recursive_definition
    false
    cc_env
    'I_last_in_E'
    "I_last_in_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = I_last_in_";;

let I_srdy_in_E = new_recursive_definition
    false
    cc_env
    'I_srdy_in_E'
    "I_srdy_in_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = I_srdy_in_";;

let I_lock_E = new_recursive_definition
    false
    cc_env
    'I_lock_E'
    "I_lock_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = I_lock_";;

let I_cale_E = new_recursive_definition
    false
    cc_env
    'I_cale_E'
    "I_cale_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = I_cale_";;

let I_hlda_E = new_recursive_definition
    false
    cc_env
    'I_hlda_E'
    "I_hlda_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
```

```
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
     = I_hlda_";;

let I_crqt_E = new_recursive_definition
    false
    cc_env
    'I_crqt_E'
    "I_crqt_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                    I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
     = I_crqt_";;

let CB_rqt_in_E = new_recursive_definition
    false
    cc_env
    'CB_rqt_in_E'
    "CB_rqt_in_E (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                    I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
     = CB_rqt_in_";;

let CB_ad_inE = new_recursive_definition
    false
    cc_env
    'CB_ad_inE'
    "CB_ad_inE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                    I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
     = CB_ad_in";;

let CB_ms_inE = new_recursive_definition
    false
    cc_env
    'CB_ms_inE'
    "CB_ms_inE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                    I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
     = CB_ms_in";;

let CB_ss_inE = new_recursive_definition
    false
    cc_env
    'CB_ss_inE'
    "CB_ss_inE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                    I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
     = CB_ss_in";;

let RstE = new_recursive_definition
    false
    cc_env
    'RstE'
    "RstE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                    I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
     = Rst";;

let ClkDE = new_recursive_definition
    false
    cc_env
    'ClkDE'
    "ClkDE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                    I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                    CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                    Pmm_failure Piu_invalid Ccr Reset_error)
```

```
     = ClkD";;

let IdE = new_recursive_definition
    false
    cc_env
    'IdE'
    "IdE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = Id";;

let ChannelIDE = new_recursive_definition
    false
    cc_env
    'ChannelIDE'
    "ChannelIDE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = ChannelID";;

let Pmm_failureE = new_recursive_definition
    false
    cc_env
    'Pmm_failureE'
    "Pmm_failureE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = Pmm_failure";;

let Piu_invalidE = new_recursive_definition
    false
    cc_env
    'Piu_invalidE'
    "Piu_invalidE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = Piu_invalid";;

let CcrE = new_recursive_definition
    false
    cc_env
    'CcrE'
    "CcrE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = Ccr";;

let Reset_errorE = new_recursive_definition
    false
    cc_env
    'Reset_errorE'
    "Reset_errorE (CCEnv I_ad_in I_be_in_ I_mrdy_in_ I_rale_in_ I_male_in_
                  I_last_in_ I_srdy_in_ I_lock_ I_cale_ I_hlda_ I_crqt_
                  CB_rqt_in_  CB_ad_in CB_ms_in CB_ss_in Rst ClkD Id ChannelID
                  Pmm_failure Piu_invalid Ccr Reset_error)
     = Reset_error";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm cc_env);;

let CCEnv_Selectors_Work = prove_thm
   ('CCEnv_Selectors_Work',
    "! e:cc_env .
     e = (CCEnv (I_ad_inE e) (I_be_in_E e) (I_mrdy_in_E e) (I_rale_in_E e)
               (I_male_in_E e) (I_last_in_E e) (I_srdy_in_E e) (I_lock_E e)
               (I_cale_E e) (I_hlda_E e) (I_crqt_E e) (CB_rqt_in_E e)
               (CB_ad_inE e) (CB_ms_inE e) (CB_ss_inE e) (RstE e) (ClkDE e)
```

```
            (IdE e) (ChannelIDE e) (Pmm_failureE e) (Piu_invalidE e) (CcrE e)
            (Reset_errorE e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:cc_env" Env_CASES)
    THEN REWRITE_TAC [I_ad_inE; I_be_in_E; I_mrdy_in_E; I_rale_in_E;
                      I_male_in_E; I_last_in_E; I_srdy_in_E; I_lock_E; I_cale_E;
                      I_hlda_E; I_crqt_E; CB_rqt_in_E;  CB_ad_inE; CB_ms_inE;
                      CB_ss_inE; RstE; ClkDE; IdE; ChannelIDE; Pmm_failureE;
                      Piu_invalidE; CcrE; Reset_errorE]
  );;

%-----------------------------------------------------------------------------
   Abstract data type for the output.
-----------------------------------------------------------------------------%

let cc_out =
   define_type 'cc_out'
               'cc_out = CCOut bool#bool wire#wire bool#bool wire#wire wire#wire
                              wire#wire wire#wire busn#busn busn#busn
                              bool#bool wordn#wordn wordn#wordn busn#busn
                              wordn#wordn bool#bool bool#bool';;

let I_cgnt_O = new_recursive_definition
    false
    cc_out
    'I_cgnt_O'
    "I_cgnt_O (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                     I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                     CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                     CB_parity)
     = I_cgnt_";;

let I_mrdy_out_O = new_recursive_definition
    false
    cc_out
    'I_mrdy_out_O'
    "I_mrdy_out_O (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                     I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                     CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                     CB_parity)
     = I_mrdy_out_";;

let I_hold_O = new_recursive_definition
    false
    cc_out
    'I_hold_O'
    "I_hold_O (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                     I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                     CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                     CB_parity)
     = I_hold_";;

let I_rale_out_O = new_recursive_definition
    false
    cc_out
    'I_rale_out_O'
    "I_rale_out_O (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                     I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                     CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                     CB_parity)
     = I_rale_out_";;

let I_male_out_O = new_recursive_definition
    false
    cc_out
    'I_male_out_O'
    "I_male_out_O (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                     I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                     CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                     CB_parity)
     = I_male_out_";;
```

```
let I_last_out_0 = new_recursive_definition
    false
    cc_out
    'I_last_out_0'
    "I_last_out_0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                   I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                   CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                   CB_parity)
     = I_last_out_";;

let I_srdy_out_0 = new_recursive_definition
    false
    cc_out
    'I_srdy_out_0'
    "I_srdy_out_0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                   I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                   CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                   CB_parity)
     = I_srdy_out_";;

let I_ad_out0 = new_recursive_definition
    false
    cc_out
    'I_ad_out0'
    "I_ad_out0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                   I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                   CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                   CB_parity)
     = I_ad_out";;

let I_be_out_0 = new_recursive_definition
    false
    cc_out
    'I_be_out_0'
    "I_be_out_0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                   I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                   CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                   CB_parity)
     = I_be_out_";;

let CB_rqt_out_0 = new_recursive_definition
    false
    cc_out
    'CB_rqt_out_0'
    "CB_rqt_out_0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                   I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                   CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                   CB_parity)
     = CB_rqt_out_";;

let CB_ms_out0 = new_recursive_definition
    false
    cc_out
    'CB_ms_out0'
    "CB_ms_out0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                   I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                   CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                   CB_parity)
     = CB_ms_out";;

let CB_ss_out0 = new_recursive_definition
    false
    cc_out
    'CB_ss_out0'
    "CB_ss_out0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                   I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                   CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                   CB_parity)
     = CB_ss_out";;

let CB_ad_out0 = new_recursive_definition
    false
```

```
    cc_out
    'CB_ad_out0'
    "CB_ad_out0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                    I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                    CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                    CB_parity)
     = CB_ad_out";;

let C_ss_out0 = new_recursive_definition
    false
    cc_out
    'C_ss_out0'
    "C_ss_out0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                    I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                    CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                    CB_parity)
     = C_ss_out";;

let Disable_writes0 = new_recursive_definition
    false
    cc_out
    'Disable_writes0'
    "Disable_writes0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                    I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                    CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                    CB_parity)
     = Disable_writes";;

let CB_parity0 = new_recursive_definition
    false
    cc_out
    'CB_parity0'
    "CB_parity0 (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_
                    I_last_out_ I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_
                    CB_ms_out CB_ss_out CB_ad_out C_ss_out Disable_writes
                    CB_parity)
     = CB_parity";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm cc_out);;

let CCOut_Selectors_Work = prove_thm
   ('CCOut_Selectors_Work',
    "! p:cc_out .
     p = (CCOut (I_cgnt_0 p) (I_mrdy_out_0 p) (I_hold_0 p) (I_rale_out_0 p)
              (I_male_out_0 p) (I_last_out_0 p) (I_srdy_out_0 p) (I_ad_out0 p)
              (I_be_out_0 p) (CB_rqt_out_0 p) (CB_ms_out0 p) (CB_ss_out0 p)
              (CB_ad_out0 p) (C_ss_out0 p) (Disable_writes0 p) (CB_parity0 p))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "p:cc_out" Out_CASES)
     THEN REWRITE_TAC [I_cgnt_0; I_mrdy_out_0; I_hold_0; I_rale_out_0;
                       I_male_out_0; I_last_out_0; I_srdy_out_0; I_ad_out0;
                       I_be_out_0; CB_rqt_out_0; CB_ms_out0; CB_ss_out0;
                       CB_ad_out0; C_ss_out0; Disable_writes0; CB_parity0]
   );;

close_theory();;



%-----------------------------------------------------------------------------

   File:        cblock_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        3 March 1993

   This file contains the ml source for the gate-level specification of the
   C-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.
```

```
-------------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftop/piu/hol/cport/';
                                  '/home/elvis6/dfura/ftop/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/ml/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing', true);;

system 'rm cblock_def.th';;

new_theory 'cblock_def';;

loadf 'abs_theory';;

loadf 'aux_defs';;

map new_parent ['counters_def';'caux_def';'wordn_def';'array_def';'ineq'];;
map load_parent ['piuaux_def';'gates_def1';'latches_def';'ffs_def';'cfsms_def'];;


let MSTART = "WORDN 2 4";;
let MEND = "WORDN 2 5";;
let MRDY = "WORDN 2 6";;
let MWAIT = "WORDN 2 7";;
let MABORT = "WORDN 2 0";;

let SACK = "WORDN 2 5";;
let SRDY = "WORDN 2 6";;
let SWAIT = "WORDN 2 7";;
let SABORT = "WORDN 2 0";;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%-------------------------------------------------------------------------------
Input logic for C_last_in_ flip-flop.
-------------------------------------------------------------------------------%

let Last_Logic_GATE = new_definition
   ('Last_Logic_GATE',
    "! (rst clkD mfsm_md1 mfsm_mabort last_in_inE :time->bool#bool) .
     Last_Logic_GATE rst clkD mfsm_md1 mfsm_mabort last_in_inE =
        ! t:time .
        last_in_inE t =
           ((ASel(rst t) \/
             (ASel(clkD t) /\ ASel(mfsm_md1 t)) \/
             ASel(mfsm_mabort t)),
            (BSel(rst t) \/
             (BSel(clkD t) /\ BSel(mfsm_md1 t)) \/
             BSel(mfsm_mabort t)))"
   );;

%-------------------------------------------------------------------------------
Input logic for C_last_out_ latch.
-------------------------------------------------------------------------------%

let Hold_Logic_GATE = new_definition
   ('Hold_Logic_GATE',
    "! (cb_ms :time->wordn#wordn)
       (clkD sfsm_sa1 last_out_inS last_out_inR last_out_inE :time->bool#bool) .
     Hold_Logic_GATE cb_ms clkD sfsm_sa1 last_out_inS last_out_inR last_out_inE =
        ! t:time .
        (last_out_inS t = ((ASel(sfsm_sa1 t)), (BSel(sfsm_sa1 t)))) /\
        (last_out_inR t =
           ((ASel(clkD t) /\ ((ASel(cb_ms t) = ^MEND) \/
                              (ASel(cb_ms t) = ^MABORT))),
            (BSel(clkD t) /\ ((BSel(cb_ms t) = ^MEND) \/
                              (BSel(cb_ms t) = ^MABORT))))) /\
        (last_out_inE t = ((ASel(last_out_inS t) \/ ASel(last_out_inR t)),
                           (BSel(last_out_inS t) \/ BSel(last_out_inR t))))"
```

```
   );;

%-----------------------------------------------------------------------------
Generation logic for cout_sel signal.
-----------------------------------------------------------------------------%

let Cout_Sel_Logic_GATE = new_definition
   ('Cout_Sel_Logic_GATE',
    "! (sfsm_s_cout_sel0 mfsm_m_cout_sel1 mfsm_m_cout_sel0 :time->bool#bool)
       (sfsm_sd0 sfsm_sd1 :time->bool#bool)
       (cout_sel :time->wordn#wordn) .
     Cout_Sel_Logic_GATE sfsm_s_cout_sel0 mfsm_m_cout_sel1 mfsm_m_cout_sel0
                         sfsm_sd0 sfsm_sd1 cout_sel =
        ! t:time .
        cout_sel t =
        (((ASel(sfsm_sd0 t) \/ ASel(sfsm_sd1 t))
           => (let a0 = (ALTER ARBN 0 (ASel(sfsm_s_cout_sel0 t)))
               in       (ALTER a0 1 F))
           |  (let a0 = (ALTER ARBN 0 (ASel(mfsm_m_cout_sel0 t)))
               in       (ALTER a0 1 (ASel(mfsm_m_cout_sel1 t))))),
         ((BSel(sfsm_sd0 t) \/ BSel(sfsm_sd1 t))
           => (let b0 = (ALTER ARBN 0 (BSel(sfsm_s_cout_sel0 t)))
               in       (ALTER b0 1 F))
           |  (let b0 = (ALTER ARBN 0 (BSel(mfsm_m_cout_sel0 t)))
               in       (ALTER b0 1 (BSel(mfsm_m_cout_sel1 t))))))"
   );;

%-----------------------------------------------------------------------------
Generation logic for srdy signal.
-----------------------------------------------------------------------------%

let Srdy_In_Logic_GATE = new_definition
   ('Srdy_In_Logic_GATE',
    "! (cb_ss :time->wordn#wordn) (dfsm_srdy :time->bool#bool) .
     Srdy_In_Logic_GATE cb_ss dfsm_srdy =
        ! t:time .
        dfsm_srdy t = ((ASel(cb_ss t) = ^SRDY), (BSel(cb_ss t) = ^SRDY))"
   );;

%-----------------------------------------------------------------------------
Input logic for C_wrdy, C_rrdy latches.
-----------------------------------------------------------------------------%

let Rdy_Logic_GATE = new_definition
   ('Rdy_Logic_GATE',
    "! (mfsm_md0 mfsm_md1 clkD write srdy wrdy_inD rrdy_inD :time->bool#bool) .
     Rdy_Logic_GATE mfsm_md0 mfsm_md1 clkD write srdy wrdy_inD rrdy_inD =
        ! t:time .
        (wrdy_inD t =
           ((ASel(srdy t) /\ ASel(write t) /\ ASel(mfsm_md1 t) /\
             ASel(clkD t)),
            (BSel(srdy t) /\ BSel(write t) /\ BSel(mfsm_md1 t) /\
             BSel(clkD t)))) /\
        (rrdy_inD t =
           ((ASel(srdy t) /\ ~ASel(write t) /\ ASel(mfsm_md0 t) /\
             ASel(clkD t)),
            (BSel(srdy t) /\ ~BSel(write t) /\ BSel(mfsm_md0 t) /\
             BSel(clkD t))))"
   );;

%-----------------------------------------------------------------------------
Generation logic for I_srdy_out_ signal.
-----------------------------------------------------------------------------%

let ISrdy_Out_Logic_GATE = new_definition
   ('ISrdy_Out_Logic_GATE',
    "! (wrdy_outQ rrdy_outQ fsm_mabort cale_ srdy_en :time->bool#bool)
       (isrdy_inD isrdy_inE :time->bool#bool) .
     ISrdy_Out_Logic_GATE wrdy_outQ rrdy_outQ fsm_mabort cale_ srdy_en
                          isrdy_inD isrdy_inE =
        ! t:time .
        (isrdy_inD t =
```

```
          ((~(ASel(wrdy_outQ t) \/ ASel(rrdy_outQ t) \/ ASel(fsm_mabort t))),
           (~(BSel(wrdy_outQ t) \/ BSel(rrdy_outQ t) \/ BSel(fsm_mabort t)))))) /\
        (isrdy_inE t =
           ((~ASel(cale_ t) \/ ASel(srdy_en t)),
            (~BSel(cale_ t) \/ BSel(srdy_en t))))"
   );;

%-------------------------------------------------------------------------------
Generation logic for CBss_out signal.
-------------------------------------------------------------------------------%

let CBss_Out_Logic_GATE = new_definition
   ('CBss_Out_Logic_GATE',
    "! (sfsm_ss cbss_out :time->wordn#wordn)
       (pmm_failure piu_valid :time->bool#bool) .
     CBss_Out_Logic_GATE sfsm_ss pmm_failure piu_valid cbss_out =
        ! t:time .
       cbss_out t =
          ((let a1_0 = (MALTER
                          ARBN
                          (1,0)
                          (SUBARRAY (ASel(sfsm_ss t)) (1,0)))
            in          (ALTER
                          a1_0
                          (2)
                          ((ELEMENT (ASel(sfsm_ss t)) (2)) /\
                           ~ASel(pmm_failure t) /\ ~ASel(piu_valid t)))),
           (let b1_0 = (MALTER
                          ARBN
                          (1,0)
                          (SUBARRAY (BSel(sfsm_ss t)) (1,0)))
            in          (ALTER
                          b1_0
                          (2)
                          ((ELEMENT (BSel(sfsm_ss t)) (2)) /\
                           ~BSel(pmm_failure t) /\ ~BSel(piu_valid t)))))"
   );;

%-------------------------------------------------------------------------------
Generation logic for CBms_out signal.
-------------------------------------------------------------------------------%

let CBms_Out_Logic_GATE = new_definition
   ('CBms_Out_Logic_GATE',
    "! (mfsm_ms cbms_out :time->wordn#wordn)
       (pmm_failure piu_valid :time->bool#bool) .
     CBms_Out_Logic_GATE mfsm_ms pmm_failure piu_valid cbms_out =
        ! t:time .
        cbms_out t =
        ((let a1_0 = (MALTER
                        ARBN
                        (1,0)
                        (SUBARRAY (ASel(mfsm_ms t)) (1,0)))
          in          (ALTER
                        a1_0
                        (2)
                        ((ELEMENT (ASel(mfsm_ms t)) (2)) /\
                         ~ASel(pmm_failure t) /\ ~ASel(piu_valid t)))),
         (let b1_0 = (MALTER
                        ARBN
                        (1,0)
                        (SUBARRAY (BSel(mfsm_ms t)) (1,0)))
          in          (ALTER
                        b1_0
                        (2)
                        ((ELEMENT (BSel(mfsm_ms t)) (2)) /\
                         ~BSel(pmm_failure t) /\ ~BSel(piu_valid t)))))"
   );;

%-------------------------------------------------------------------------------
Generation logic for cout_1_le signal.
-------------------------------------------------------------------------------%
```

```
let Cout_1_Le_Logic_GATE = new_definition
   ('Cout_1_Le_Logic_GATE',
    "! (dfsm_master cout_0_le_del dfsm_cout_1_le cout_1_le :time->bool#bool) .
     Cout_1_Le_Logic_GATE dfsm_master cout_0_le_del dfsm_cout_1_le cout_1_le =
        ! t:time .
        cout_1_le t =
           (((ASel(dfsm_cout_1_le t) /\ ~ASel(dfsm_master t)) \/
             (ASel(dfsm_master t) /\ ASel(cout_0_le_del t))),
            ((BSel(dfsm_cout_1_le t) /\ ~BSel(dfsm_master t)) \/
             (BSel(dfsm_master t) /\ BSel(cout_0_le_del t))))"
   );;

%-----------------------------------------------------------------------------
Generation logic for iad_en signal.
-----------------------------------------------------------------------------%

let Iad_En_Logic_GATE = new_definition
   ('Iad_En_Logic_GATE',
    "! (mfsm_iad_en_m sfsm_iad_en_s iad_en_s_del iad_en :time->bool#bool) .
     Iad_En_Logic_GATE mfsm_iad_en_m sfsm_iad_en_s iad_en_s_del iad_en =
        ! t:time .
        iad_en t =
           ((ASel(mfsm_iad_en_m t) \/
             ASel(sfsm_iad_en_s t) \/
             ASel(iad_en_s_del t)),
            (BSel(mfsm_iad_en_m t) \/
             BSel(sfsm_iad_en_s t) \/
             BSel(iad_en_s_del t)))"
   );;

%-----------------------------------------------------------------------------
Generation logic for c_pe_cnt signal.
-----------------------------------------------------------------------------%

let Pe_Cnt_Logic_GATE = new_definition
   ('Pe_Cnt_Logic_GATE',
    "! (cb_ss_in :time->wordn#wordn)
       (clkD sfsm_sparity mfsm_mparity c_pe_cnt :time->bool#bool) .
     Pe_Cnt_Logic_GATE clkD sfsm_sparity mfsm_mparity cb_ss_in c_pe_cnt =
        ! t:time .
        c_pe_cnt t =
           ((ASel(clkD t) /\
             (~(ASel(sfsm_sparity t) = ASel(mfsm_mparity t)) \/
              ((SUBARRAY (ASel(cb_ss_in t)) (1,0)) = WORDN 2 0))),
            (BSel(clkD t) /\
             (~(BSel(sfsm_sparity t) = BSel(mfsm_mparity t)) \/
              ((SUBARRAY (BSel(cb_ss_in t)) (1,0)) = WORDN 2 0))))"
   );;

%-----------------------------------------------------------------------------
Generation logic for c_grant, c_busy signals.
-----------------------------------------------------------------------------%

let Grant_Logic_GATE = new_definition
   ('Grant_Logic_GATE',
    "! (id rqt_ :time->wordn#wordn) (busy grant :time->bool#bool) .
     Grant_Logic_GATE id rqt_ busy grant =
        ! t:time .
        (busy t =
           ((~((SUBARRAY (ASel(rqt_ t)) (3,1)) = WORDN 2 7)),
            (~((SUBARRAY (BSel(rqt_ t)) (3,1)) = WORDN 2 7)))) /\
        (grant t =
           (((((SUBARRAY (ASel(id t)) (1,0)) = WORDN 1 0) /\
                         ~(ELEMENT (ASel(rqt_ t)) (0))    ) \/
             (((SUBARRAY (ASel(id t)) (1,0)) = WORDN 1 1) /\
                         ~(ELEMENT (ASel(rqt_ t)) (0)) /\
                          (ELEMENT (ASel(rqt_ t)) (1))    ) \/
             (((SUBARRAY (ASel(id t)) (1,0)) = WORDN 1 2) /\
                         ~(ELEMENT (ASel(rqt_ t)) (0)) /\
                          (ELEMENT (ASel(rqt_ t)) (1)) /\
                          (ELEMENT (ASel(rqt_ t)) (2))    ) \/
```

```
            (((SUBARRAY (ASel(id t)) (1,0)) = WORDN 1 3) /\
                        ~(ELEMENT (ASel(rqt_ t)) (0)) /\
                         (ELEMENT (ASel(rqt_ t)) (1)) /\
                         (ELEMENT (ASel(rqt_ t)) (2)) /\
                         (ELEMENT (ASel(rqt_ t)) (3))   ),
           ((((SUBARRAY (BSel(id t)) (1,0)) = WORDN 1 0) /\
                        ~(ELEMENT (BSel(rqt_ t)) (0))   ) \/
            (((SUBARRAY (BSel(id t)) (1,0)) = WORDN 1 1) /\
                        ~(ELEMENT (BSel(rqt_ t)) (0)) /\
                         (ELEMENT (BSel(rqt_ t)) (1))   ) \/
            (((SUBARRAY (BSel(id t)) (1,0)) = WORDN 1 2) /\
                        ~(ELEMENT (BSel(rqt_ t)) (0)) /\
                         (ELEMENT (BSel(rqt_ t)) (1)) /\
                         (ELEMENT (BSel(rqt_ t)) (2))   ) \/
            (((SUBARRAY (BSel(id t)) (1,0)) = WORDN 1 3) /\
                        ~(ELEMENT (BSel(rqt_ t)) (0)) /\
                         (ELEMENT (BSel(rqt_ t)) (1)) /\
                         (ELEMENT (BSel(rqt_ t)) (2)) /\
                         (ELEMENT (BSel(rqt_ t)) (3))   )    )))"
   );;

%-------------------------------------------------------------------------------
Generation logic for addressed signal.
-------------------------------------------------------------------------------%

let Addressed_Logic_GATE = new_definition
   ('Addressed_Logic_GATE',
    "! (id source :time->wordn#wordn) (addressed :time->bool#bool) .
     Addressed_Logic_GATE id source addressed =
        ! t:time .
        addressed t =
           ((ASel(id t) = (SUBARRAY (ASel(source t)) (15,10))),
            (BSel(id t) = (SUBARRAY (BSel(source t)) (15,10))))"
   );;

%-------------------------------------------------------------------------------
Generation logic for Disable_writes signal.
-------------------------------------------------------------------------------%

let D_Writes_Logic_GATE = new_definition
   ('D_Writes_Logic_GATE',
    "! (chan_id source :time->wordn#wordn)
       (dfsm_slave disable_writes :time->bool#bool) .
     D_Writes_Logic_GATE dfsm_slave chan_id source disable_writes =
       ! t:time .
       disable_writes t =
          ((ASel(dfsm_slave t) /\
           (~(ELEMENT
                (SUBARRAY (ASel(source t)) (9,6))
                (VAL 1 (ASel(chan_id t))))))),
          ((BSel(dfsm_slave t) /\
           (~(ELEMENT
                (SUBARRAY (BSel(source t)) (9,6))
                (VAL 1 (BSel(chan_id t))))))))"
   );;

%-------------------------------------------------------------------------------
Generation logic for c_pe signal.
-------------------------------------------------------------------------------%

let Parity_Decode_Logic_GATE = new_definition
   ('Parity_Decode_Logic_GATE',
    "! (rep :^REP_ty) (cad_in_det :time->bool#bool)
       (cad_in cad_in_dec :time->wordn#wordn) .
     Parity_Decode_Logic_GATE rep cad_in cad_in_dec cad_in_det =
        ! t:time .
        (cad_in_dec t =
           ((Par_Dec rep (ASel(cad_in t))),
            (Par_Dec rep (BSel(cad_in t))))) /\
        (cad_in_det t =
           ((Par_Det rep (ASel(cad_in t))),
            (Par_Det rep (BSel(cad_in t)))))"
```

```
   );;

%-----------------------------------------------------------------------------
Input logic for C_parity latch.
-----------------------------------------------------------------------------%

let Parity_Signal_Inputs_GATE = new_definition
   ('Parity_Signal_Inputs_GATE',
    "! (rst cad_in_det clkD c_pe_cnt reset_error :time->bool#bool)
       (c_parity_inS c_parity_inR c_parity_inE :time->bool#bool) .
     Parity_Signal_Inputs_GATE rst cad_in_det clkD c_pe_cnt reset_error
                               c_parity_inS c_parity_inR c_parity_inE =
        ! t:time .
        (c_parity_inS t =
           ((ASel(cad_in_det t) /\ ASel(clkD t) /\ ASel(c_pe_cnt t)),
            (BSel(cad_in_det t) /\ BSel(clkD t) /\ BSel(c_pe_cnt t)))) /\
        (c_parity_inR t =
           ((ASel(rst t) \/ ASel(reset_error t)),
            (BSel(rst t) \/ BSel(reset_error t)))) /\
        (c_parity_inE t =
           ((ASel(c_parity_inS t) \/ ASel(c_parity_inR t)),
            (BSel(c_parity_inS t) \/ BSel(c_parity_inR t))))"
   );;

%-----------------------------------------------------------------------------
C-Bus input latches.
-----------------------------------------------------------------------------%

let CB_In_Latches_GATE = new_definition
   ('CB_In_Latches_GATE',
    "! (cad_in_dec source sizewrbe iad_preout :time->wordn#wordn)
       (c_source c_data_in c_sizewrbe c_iad_preout :time->wordn)
       (rst cin_0_le cin_1_le cin_2_le cin_3_le cin_4_le :time->bool#bool) .
     CB_In_Latches_GATE rst cad_in_dec cin_0_le cin_1_le cin_2_le cin_3_le
                        cin_4_le source sizewrbe iad_preout c_source
                        c_data_in c_sizewrbe c_iad_preout =
        ! t:time .
        (c_source (t+1) =
           (BSel(rst t)) => WORDN 15 0 |
           (BSel(cin_3_le t)) => (BSel(cad_in_dec t)) | (c_source t)) /\
        (let a31_16 =
           (MALTER
              (c_data_in t)
              (31,16)
              ((BSel(rst t)) => WORDN 15 0 |
               (BSel(cin_1_le t))
                  => (BSel(cad_in_dec t))
                  |  (SUBARRAY (c_data_in t) (31,16)))) in
         let a31_0 =
           (MALTER
              a31_16
              (15,0)
              ((BSel(rst t)) => WORDN 15 0 |
               (BSel(cin_0_le t))
                  => (BSel(cad_in_dec t))
                  |  (SUBARRAY (c_data_in t) (15,0)))) in
        (c_data_in (t+1) = a31_0)) /\
        (c_sizewrbe (t+1) =
           (BSel(rst t)) => WORDN 9 0 |
           (BSel(cin_4_le t)) => (SUBARRAY (c_data_in t) (31,22))
                              |  (c_sizewrbe t)) /\
        (c_iad_preout (t+1) =
           (ASel(cin_2_le t)) => (c_data_in t) | (c_iad_preout t)) /\
        (source t = ((c_source t), (c_source (t+1)))) /\
        (sizewrbe t = ((c_sizewrbe t), (c_sizewrbe (t+1)))) /\
        (iad_preout t = ((c_iad_preout (t+1)), (c_iad_preout (t+1))))"
   );;

%-----------------------------------------------------------------------------
Generation logic for I_be_out_ signal.
-----------------------------------------------------------------------------%
```

```
let BE_Out_Logic_GATE = new_definition
   ('BE_Out_Logic_GATE',
    "! (sizewrbe :time->wordn#wordn) (hlda_ :time->bool#bool)
       (be_out :time->busn#busn) .
     BE_Out_Logic_GATE sizewrbe hlda_ be_out =
        ! t:time .
        be_out t =
           (((~ASel(hlda_ t))
               => (BUSN (SUBARRAY (ASel(sizewrbe t)) (9,6))) | Offn),
            ((~BSel(hlda_ t))
               => (BUSN (SUBARRAY (BSel(sizewrbe t)) (9,6))) | Offn)"
   );;

%-----------------------------------------------------------------------------
Generation logic for write signal.
-----------------------------------------------------------------------------%

let Write_Logic_GATE = new_definition
   ('Write_Logic_GATE',
    "! (iad_in sizewrbe :time->wordn#wordn) (C_wr :time->bool)
       (cale_ master_tran write :time->bool#bool) .
     Write_Logic_GATE iad_in sizewrbe cale_ master_tran C_wr write =
        ! t:time .
        (C_wr (t+1) =
           (~BSel(cale_ t)) => (ELEMENT (BSel(iad_in t)) (27)) | C_wr t) /\
        (write t =
          (((ASel(master_tran t))
               => (C_wr t) | (ELEMENT (ASel(sizewrbe t)) (5))),
           ((BSel(master_tran t))
               => (C_wr (t+1)) | (ELEMENT (BSel(sizewrbe t)) (5)))))"
   );;

%-----------------------------------------------------------------------------
C-Bus output latches.
-----------------------------------------------------------------------------%

let CB_Out_Logic_GATE = new_definition
   ('CB_Out_Logic_GATE',
    "! (rep :^REP_ty)
       (iad_in ccr cout_sel cad_preout :time->wordn#wordn)
       (dfsm_cout_0_le cout_1_le mfsm_mrequest :time->bool#bool)
       (C_iad_in C_a1a0 C_a3a2 :time->wordn) .
     CB_Out_Logic_GATE rep iad_in ccr dfsm_cout_0_le cout_1_le mfsm_mrequest
                       cout_sel cad_preout C_iad_in C_a1a0 C_a3a2 =
        ! t:time .
        (C_iad_in (t+1) =
           (BSel(dfsm_cout_0_le t)) => (BSel(iad_in t)) | (C_iad_in t)) /\
        (C_a1a0 (t+1) =
           (ASel(cout_1_le t)) => (C_iad_in t) | (C_a1a0 t)) /\
        (C_a3a2 (t+1) =
           (ASel(mfsm_mrequest t)) => (ASel(ccr t)) | (C_a3a2 t)) /\
        (cad_preout t =
          (((ASel(cout_sel t) = WORDN 1 0)
               => (Par_Enc rep (SUBARRAY (C_a1a0 (t+1)) (15,0))) |
             (ASel(cout_sel t) = WORDN 1 1)
               => (Par_Enc rep (SUBARRAY (C_a1a0 (t+1)) (31,16))) |
             (ASel(cout_sel t) = WORDN 1 2)
               => (Par_Enc rep (SUBARRAY (C_a3a2 (t+1)) (15,0)))
                | (Par_Enc rep (SUBARRAY (C_a3a2 (t+1)) (31,16)))),
           ((BSel(cout_sel t) = WORDN 1 0)
               => (Par_Enc rep (SUBARRAY (C_a1a0 (t+1)) (15,0))) |
             (BSel(cout_sel t) = WORDN 1 1)
               => (Par_Enc rep (SUBARRAY (C_a1a0 (t+1)) (31,16))) |
             (BSel(cout_sel t) = WORDN 1 2)
               => (Par_Enc rep (SUBARRAY (C_a3a2 (t+1)) (15,0)))
                | (Par_Enc rep (SUBARRAY (C_a3a2 (t+1)) (31,16))))))"
   );;

%-----------------------------------------------------------------------------
C-Port Block.
-----------------------------------------------------------------------------%
```

```
let CBlock_GATE = new_definition
   ('CBlock_GATE',
    "! (rep :^REP_ty) (s :time->cc_state) (e :time->cc_env) (p :time->cc_out) .
     CBlock_GATE rep s e p =
        ? (mfsm_mabort mfsm_midle mfsm_mrequest mfsm_ma3 :time->bool#bool)
          (mfsm_ma2 mfsm_ma1 mfsm_ma0 mfsm_md1 mfsm_md0 :time->bool#bool)
          (mfsm_iad_en_m mfsm_m_cout_sel1 mfsm_m_cout_sel0 :time->bool#bool)
          (mfsm_cm_en mfsm_abort_le_en_ mfsm_mparity :time->bool#bool)
          (sfsm_iad_en_s sfsm_sidle sfsm_slock sfsm_sa1 :time->bool#bool)
          (sfsm_sa0 sfsm_sale sfsm_sd1 sfsm_sd0 sfsm_sack :time->bool#bool)
          (sfsm_sabort sfsm_s_cout_sel0 sfsm_sparity :time->bool#bool)
          (sfsm_srdy_en :time->bool#bool)
          (dfsm_srdy dfsm_master dfsm_slave dfsm_cin_0_le :time->bool#bool)
          (dfsm_cin_1_le dfsm_cin_3_le dfsm_cin_4_le :time->bool#bool)
          (dfsm_cout_0_le dfsm_cout_1_le dfsm_cad_en_ dfsm_male_:time->bool#bool)
          (dfsm_rale_ dfsm_mrdy_ :time->bool#bool)
          (lock_in_inE lock_in_outQ last_in_inE last_in_outQ :time->bool#bool)
          (clkA_outQ last_out_inS last_out_inR last_out_inE :time->bool#bool)
          (last_out_outQ hlda sstatus_en_ sidle_del_outQ :time->bool#bool)
          (mrqt_del_outQ mstatus_en_ write wrdy_inD :time->bool#bool)
          (rrdy_inD wrdy_outQ rrdy_outQ isrdy_inD isrdy_inE :time->bool#bool)
          (cout_0_le_del_out cin_2_le cout_1_le :time->bool#bool)
          (mrdy_del_out iad_en_s_del_outQ iad_en c_pe_cnt busy :time->bool#bool)
          (grant addressed cad_in_det c_parity_inS c_parity_inR :time->bool#bool)
          (c_parity_inE :time->bool#bool)
          (mfsm_ms sfsm_ss cout_sel cad_in_dec source :time->wordn#wordn)
          (sizewrbe iad_preout cad_preout :time->wordn#wordn) .
        (OR2_GATE (sig RstE e) mfsm_ma1 lock_in_inE) /\
        (DRELatB_GATE (sig I_lock_E e) (sig RstE e) lock_in_inE
                      (sig C_lock_in_S s) lock_in_outQ) /\
        (Last_Logic_GATE (sig RstE e) (sig ClkDE e) mfsm_md1 mfsm_mabort
                         last_in_inE) /\
        (DREFFB_GATE (sig I_last_in_E e) last_in_inE (sig RstE e)
                     (sig C_last_in_S s) last_in_outQ) /\
        (DEFFnB_GATE (sig CB_ss_inE e) mfsm_abort_le_en_ (sig C_ssS s)
                     (sig C_ss_outO p)) /\
        (DFFA_GATE (sig ClkDE e) (sig C_clkAS s) clkA_outQ) /\
        (Hold_Logic_GATE (sig CB_ms_inE e) (sig ClkDE e) sfsm_sa1 last_out_inS
                         last_out_inR last_out_inE) /\
        (DSRELatB_GATE GND last_out_inS last_out_inR last_out_inE
                       (sig C_last_out_S s) last_out_outQ) /\
        (TRIBUF_GATE last_out_outQ hlda (sig I_last_out_O p)) /\
        (OR2_GATE sfsm_sidle sfsm_sabort sstatus_en_) /\
        (DFFA_GATE sfsm_sidle (sig C_sidle_delS s) sidle_del_outQ) /\
        (DFFA_GATE mfsm_mrequest (sig C_mrqt_delS s) mrqt_del_outQ) /\
        (Cout_Sel_Logic_GATE sfsm_s_cout_sel0 mfsm_m_cout_sel1 mfsm_m_cout_sel0
                             sfsm_sd0 sfsm_sd1 cout_sel) /\
        (NOT_GATE mfsm_cm_en mstatus_en_) /\
        (DEFFA_GATE sfsm_sidle (sig ClkDE e) (sig C_hold_S s)
                    (sig I_hold_O p)) /\
        (Srdy_In_Logic_GATE (sig CB_ss_inE e) dfsm_srdy) /\
        (Rdy_Logic_GATE mfsm_md0 mfsm_md1 (sig ClkDE e) write dfsm_srdy wrdy_inD
                        rrdy_inD) /\
        (DFFA_GATE wrdy_inD (sig C_wrdyS s) wrdy_outQ) /\
        (DFFA_GATE rrdy_inD (sig C_rrdyS s) rrdy_outQ) /\
        (ISrdy_Out_Logic_GATE wrdy_outQ rrdy_outQ mfsm_mabort (sig I_cale_E e)
                              efsm_srdy_en isrdy_inD isrdy_inE) /\
        (TRIBUF_GATE isrdy_inD isrdy_inE (sig I_srdy_out_O p)) /\
        (CBss_Out_Logic_GATE sfsm_ss (sig Pmm_failureE e) (sig Piu_invalidE e)
                             (sig CB_ss_outO p)) /\
        (DFFA_GATE dfsm_cout_0_le (sig C_cout_0_le_delS s) cout_0_le_del_out) /\
        (DFFA_GATE dfsm_cin_0_le (sig C_cin_2_leS s) cin_2_le) /\
        (Cout_1_Le_Logic_GATE dfsm_master cout_0_le_del_out dfsm_cout_1_le
                              cout_1_le) /\
        (DFFA_GATE dfsm_mrdy_ (sig C_mrdy_del_S s) mrdy_del_out) /\
        (NOT_GATE (sig I_hlda_E e) hlda) /\
        (TRIBUF_GATE dfsm_male_ hlda (sig I_male_out_O p)) /\
        (TRIBUF_GATE dfsm_rale_ hlda (sig I_rale_out_O p)) /\
        (TRIBUF_GATE mrdy_del_out hlda (sig I_mrdy_out_O p)) /\
        (DEFFA_GATE sfsm_iad_en_s (sig ClkDE e) (sig C_iad_en_s_delS s)
                    iad_en_s_del_outQ) /\
        (Iad_En_Logic_GATE mfsm_iad_en_m sfsm_iad_en_s iad_en_s_del_outQ
```

```
                        iad_en) /\
        (CBms_Out_Logic_GATE mfsm_ms (sig Pmm_failureE e) (sig Piu_invalidE e)
                             (sig CB_ms_outO p)) /\
        (Pe_Cnt_Logic_GATE (sig ClkDE e) sfsm_sparity mfsm_mparity
                           (sig CB_ss_inE e) c_pe_cnt) /\
        (Grant_Logic_GATE (sig IdE e) (sig CB_rqt_in_E e) busy grant) /\
        (Addressed_Logic_GATE (sig IdE e) source addressed) /\
        (D_Writes_Logic_GATE dfsm_slave (sig ChannelIDE e) source
                             (sig Disable_writesO p)) /\
        (Parity_Decode_Logic_GATE rep (sig CB_ad_inE e) cad_in_dec cad_in_det) /\
        (Parity_Signal_Inputs_GATE (sig RstE e) cad_in_det (sig ClkDE e) c_pe_cnt
                                   (sig Reset_errorE e)
                                   c_parity_inS c_parity_inR c_parity_inE) /\
        (DSRELatB_GATE GND c_parity_inS c_parity_inR c_parity_inE
                       (sig C_paritys s) (sig CB_parityO p)) /\
        (CB_In_Latches_GATE (sig RstE e) cad_in_dec dfsm_cin_0_le dfsm_cin_1_le
                            cin_2_le dfsm_cin_3_le dfsm_cin_4_le source sizewrbe
                            iad_preout (sig C_sourceS s) (sig C_data_inS s)
                            (sig C_sizewrbeS s) (sig C_iad_outS s)) /\
        (BE_Out_Logic_GATE sizewrbe (sig I_hlda_E e) (sig I_be_out_O p)) /\
        (TRIBUFn_GATE iad_preout iad_en (sig I_ad_outO p)) /\
        (Write_Logic_GATE (sig I_ad_inE e) sizewrbe (sig I_cale_E e) mfsm_cm_en
                          (sig C_wrS s) write) /\
        (CB_Out_Logic_GATE rep (sig I_ad_inE e) (sig CcrE e) dfsm_cout_0_le
                           cout_1_le mfsm_mrequest cout_sel cad_preout
                           (sig C_iad_inS s) (sig C_a1a0S s) (sig C_a3a2S s)) /\
        (TRINBUFn_GATE cad_preout dfsm_cad_en_ (sig CB_ad_outO p)) /\
        (CMFSM_GATE efsm_srdy_en (sig ClkDE e) grant (sig RstE e) busy write
                    (sig I_crqt_E e) (sig I_hold_O p) last_in_outQ lock_in_outQ
                    (sig CB_ss_inE e) (sig Piu_invalidE e) (sig C_mfsm_stateS s)
                    (sig C_mfsm_srdy_enS s) (sig C_mfsm_DS s)
                    (sig C_mfsm_grantS s) (sig C_mfsm_rstS s)
                    (sig C_mfsm_busyS s) (sig C_mfsm_writeS s)
                    (sig C_mfsm_crqt_S s) (sig C_mfsm_hold_S s)
                    (sig C_mfsm_last_S s) (sig C_mfsm_lock_S s)
                    (sig C_mfsm_ssS s) (sig C_mfsm_invalidS s) mfsm_mabort
                    mfsm_midle mfsm_mrequest mfsm_ma3 mfsm_ma2 mfsm_ma1 mfsm_ma0
                    mfsm_md1 mfsm_md0 mfsm_iad_en_m mfsm_m_cout_sel1
                    mfsm_m_cout_sel0 mfsm_ms (sig CB_rqt_out_O p)
                    (sig I_cgnt_O p) mfsm_cm_en mfsm_abort_le_en_
                    mfsm_mparity) /\
        (CSFSM_GATE (sig ClkDE e) grant (sig RstE e) write addressed
                    (sig I_hlda_E e) (sig CB_ms_inE e) (sig C_sfsm_stateS s)
                    (sig C_sfsm_DS s) (sig C_sfsm_grantS s) (sig C_sfsm_rstS s)
                    (sig C_sfsm_writeS s) (sig C_sfsm_addressedS s)
                    (sig C_sfsm_hlda_S s) (sig C_sfsm_msS s) sfsm_ss
                    sfsm_iad_en_s sfsm_sidle sfsm_slock sfsm_sa1 sfsm_sa0
                    sfsm_sale sfsm_sd1 sfsm_sd0 sfsm_sack sfsm_sabort
                    sfsm_s_cout_sel0 sfsm_sparity) /\
        (CEFSM_GATE (sig I_cale_E e) (sig I_last_in_E e) (sig I_male_in_E e)
                    (sig I_rale_in_E e) (sig I_srdy_in_E e) (sig RstE e)
                    (sig C_efsm_stateS s) (sig C_efsm_cale_S s)
                    (sig C_efsm_last_S s) (sig C_efsm_male_S s)
                    (sig C_efsm_rale_S s) (sig C_efsm_srdy_S s)
                    (sig C_efsm_rstS s) efsm_srdy_en) /\
        (CDFSM_GATE dfsm_srdy (sig ClkDE e) clkA_outQ write sizewrbe sfsm_sidle
                    sidle_del_outQ sfsm_slock sfsm_sa1 sfsm_sa0 sfsm_sale
                    sfsm_sd1 sfsm_sd0 sfsm_sack mfsm_midle mrqt_del_outQ mfsm_ma3
                    mfsm_ma2 mfsm_ma1 mfsm_ma0 mfsm_md1 mfsm_md0 (sig I_cale_E e)
                    (sig I_srdy_in_E e) dfsm_master dfsm_slave dfsm_cin_0_le
                    dfsm_cin_1_le dfsm_cin_3_le dfsm_cin_4_le dfsm_cout_0_le
                    dfsm_cout_1_le dfsm_cad_en_ dfsm_male_ dfsm_rale_
                    dfsm_mrdy_)"
    );;

let CBlock_EXP = save_thm
    ('CBlock_EXP',
     (BETA_RULE
      (REWRITE_RULE [Last_Logic_GATE;Hold_Logic_GATE;
                     (EXPAND_LET_RULE Cout_Sel_Logic_GATE); Srdy_In_Logic_GATE;
                     Rdy_Logic_GATE;ISrdy_Out_Logic_GATE;
                     (EXPAND_LET_RULE CBss_Out_Logic_GATE);
```

```
                    (EXPAND_LET_RULE CBms_Out_Logic_GATE);Cout_1_Le_Logic_GATE;
                    Iad_En_Logic_GATE;Pe_Cnt_Logic_GATE;Grant_Logic_GATE;
                    Addressed_Logic_GATE;D_Writes_Logic_GATE;
                    Parity_Decode_Logic_GATE;Parity_Signal_Inputs_GATE;
                    (EXPAND_LET_RULE CB_In_Latches_GATE);BE_Out_Logic_GATE;
                    Write_Logic_GATE; CB_Out_Logic_GATE;NOT_GATE;OR2_GATE;
                    TRIBUF_GATE; TRIBUFn_GATE;DRELatB_GATE;DSRELatB_GATE;
                    DFFA_GATE;DEFFA_GATE; DEFFnB_GATE;DREFFB_GATE;
                    (EXPAND_LET_RULE CMFSM_GATE);(EXPAND_LET_RULE CSFSM_GATE);
                    CEFSM_GATE; CDFSM_GATE;ASel;BSel;GND;sig;TRINBUFn_GATE]
      (SPEC_ALL CBlock_GATE)))
   );;

close_theory();;


%-----------------------------------------------------------------------------

   File:        cclock_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        3 March 1993

   This file contains the ml source for the clock-level specification of the
   C-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.  The bulk of this code was
   translated from an M-language simulation program using a translator written by
   P.J. Windley at the University of Idaho.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/cport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

set_flag ('timing', true);;

system 'rm cclock_def.th';;

new_theory 'cclock_def';;

loadf 'abs_theory';;
loadf 'aux_defs';;

map new_parent ['piuaux_def';'caux_def';'cfsms_def';'wordn_def';'array_def'];;
map new_parent ['busn_def'];;
map new_parent ['gates_defi';'latches_def';'ffs_def';'counters_def';'ineq'];;

new_type_abbrev ('timeC',":num");;

let MSTART = "WORDN 2 4";;
let MEND = "WORDN 2 5";;
let MRDY = "WORDN 2 6";;
let MWAIT = "WORDN 2 7";;
let MABORT = "WORDN 2 0";;

let SACK = "WORDN 2 5";;
let SRDY = "WORDN 2 6";;
let SWAIT = "WORDN 2 7";;
let SABORT = "WORDN 2 0";;

let ASel = definition 'piuaux_def' 'ASel';;
let BSel = definition 'piuaux_def' 'BSel';;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%-----------------------------------------------------------------------------
   Next-state definition for C-Port instruction.
```

```
------------------------------------------------------------------------------%

let CC_NSF = new_definition
('CC_NSF',
 "! (rep :^REP_ty) (s :cc_state) (e :cc_env) .
  CC_NSF rep s e =
     let C_mfsm_state = C_mfsm_stateS s and
         C_mfsm_srdy_en = C_mfsm_srdy_enS s and
         C_mfsm_D = C_mfsm_DS s and
         C_mfsm_grant = C_mfsm_grantS s and
         C_mfsm_rst = C_mfsm_rstS s and
         C_mfsm_busy = C_mfsm_busyS s and
         C_mfsm_write = C_mfsm_writeS s and
         C_mfsm_crqt_ = C_mfsm_crqt_S s and
         C_mfsm_hold_ = C_mfsm_hold_S s and
         C_mfsm_last_ = C_mfsm_last_S s and
         C_mfsm_lock_ = C_mfsm_lock_S s and
         C_mfsm_ss = C_mfsm_ssS s and
         C_mfsm_invalid = C_mfsm_invalidS s and
         C_sfsm_state = C_sfsm_stateS s and
         C_sfsm_D = C_sfsm_DS s and
         C_sfsm_grant = C_sfsm_grantS s and
         C_sfsm_rst = C_sfsm_rstS s and
         C_sfsm_write = C_sfsm_writeS s and
         C_sfsm_addressed = C_sfsm_addressedS s and
         C_sfsm_hlda_ = C_sfsm_hlda_S s and
         C_sfsm_ms = C_sfsm_msS s and
         C_efsm_state = C_efsm_stateS s and
         C_efsm_cale_ = C_efsm_cale_S s and
         C_efsm_last_ = C_efsm_last_S s and
         C_efsm_male_ = C_efsm_male_S s and
         C_efsm_rale_ = C_efsm_rale_S s and
         C_efsm_srdy_ = C_efsm_srdy_S s and
         C_efsm_rst = C_efsm_rstS s and
         C_lock_in_ = C_lock_in_S s and
         C_last_in_ = C_last_in_S s and
         C_ss = C_ssS s and
         C_clkA = C_clkAS s and
         C_last_out_ = C_last_out_S s and
         C_sidle_del = C_sidle_delS s and
         C_mrqt_del = C_mrqt_delS s and
         C_hold_ = C_hold_S s and
         C_cout_0_le_del = C_cout_0_le_delS s and
         C_cin_2_le = C_cin_2_leS s and
         C_mrdy_del_ = C_mrdy_del_S s and
         C_iad_en_s_del = C_iad_en_s_delS s and
         C_wrdy = C_wrdyS s and
         C_rrdy = C_rrdyS s and
         C_parity = C_parityS s and
         C_source = C_sourceS s and
         C_data_in = C_data_inS s and
         C_sizewrbe = C_sizewrbeS s and
         C_iad_out = C_iad_outS s and
         C_a1a0 = C_a1a0S s and
         C_a3a2 = C_a3a2S s and
         C_iad_in = C_iad_inS s and
         C_wr = C_wrS s in
     let I_ad_in = I_ad_inE e and
         I_be_in_ = I_be_in_E e and
         I_mrdy_in_ = I_mrdy_in_E e and
         I_rale_in_ = I_rale_in_E e and
         I_male_in_ = I_male_in_E e and
         I_last_in_ = I_last_in_E e and
         I_srdy_in_ = I_srdy_in_E e and
         I_lock_ = I_lock_E e and
         I_cale_ = I_cale_E e and
         I_hlda_ = I_hlda_E e and
         I_crqt_ = I_crqt_E e and
         CB_rqt_in_ = CB_rqt_in_E e and
         CB_ad_in = CB_ad_inE e and
         CB_ms_in = CB_ms_inE e and
         CB_ss_in = CB_ss_inE e and
```

```
    Rst = RstE e and
    ClkD = ClkDE e and
    Id = IdE e and
    ChannelID = ChannelIDE e and
    Pmm_failure = Pmm_failureE e and
    Piu_invalid = Piu_invalidE e and
    Ccr = CcrE e and
    Reset_error = Reset_errorE e in
let new_C_mfsm_state =
   ((C_mfsm_rst) => CMI |
    (C_mfsm_state = CMI) =>
       ((C_mfsm_D /\ ~C_mfsm_crqt_ /\ ~C_mfsm_busy /\ ~C_mfsm_invalid)
          => CMR | CMI) |
    (C_mfsm_state = CMR) =>
       ((C_mfsm_D /\ C_mfsm_grant /\ C_mfsm_hold_) => CMA3 | CMR) |
    (C_mfsm_state = CMA3) => ((C_mfsm_D) => CMA1 | CMA3) |
    (C_mfsm_state = CMA1) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMA0 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMA1) |
    (C_mfsm_state = CMA0) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMA2 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMA0) |
    (C_mfsm_state = CMA2) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMD1 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMA2) |
    (C_mfsm_state = CMD1) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMD0 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMD1) |
    (C_mfsm_state = CMD0) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY) /\ C_mfsm_last_) => CMD1 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SRDY) /\ ~C_mfsm_last_) => CMW |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMD0) |
    (C_mfsm_state = CMW) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT |
        (C_mfsm_D /\ (C_mfsm_ss = ^SACK) /\ C_mfsm_lock_) => CMI |
        (C_mfsm_D /\ (C_mfsm_ss = ^SRDY) /\ ~C_mfsm_lock_ /\ ~C_mfsm_crqt_)
           => CMA3 | CMW) |
    (~C_mfsm_last_) => CMI | CMABT) in
let mfsm_mabort = (new_C_mfsm_state  = CMABT) in
let mfsm_midle = (new_C_mfsm_state = CMI) in
let mfsm_mrequest = (new_C_mfsm_state = CMR) in
let mfsm_ma3 = (new_C_mfsm_state = CMA3) in
let mfsm_ma2 = (new_C_mfsm_state = CMA2) in
let mfsm_ma1 = (new_C_mfsm_state = CMA1) in
let mfsm_ma0 = (new_C_mfsm_state = CMA0) in
let mfsm_md1 = (new_C_mfsm_state = CMD1) in
let mfsm_md0 = (new_C_mfsm_state = CMD0) in
let mfsm_iad_en_m =
   (((new_C_mfsm_state = CMD1) /\ ~C_mfsm_write /\ C_mfsm_srdy_en) \/
    ((new_C_mfsm_state = CMD0) /\ ~C_mfsm_write /\ C_mfsm_srdy_en) \/
    ((new_C_mfsm_state = CMW) /\ (C_mfsm_state = CMD0) /\ ~C_mfsm_write /\
     C_mfsm_srdy_en)) in
let mfsm_m_cout_sel1 =
   ((new_C_mfsm_state = CMA3) \/ (new_C_mfsm_state = CMA2)) in
let mfsm_m_cout_sel0 =
   ((new_C_mfsm_state = CMA3) \/ (new_C_mfsm_state = CMA1) \/
    (new_C_mfsm_state = CMD1)) in
let ms0 =
   (ALTER
      ARBN
      (0)
      (((new_C_mfsm_state = CMD0) /\ ~C_mfsm_last_) \/
       (new_C_mfsm_state = CMABT) \/
       ((new_C_mfsm_state = CMW) /\ C_mfsm_lock_))) in
let ms10 =
   (ALTER
      ms0
      (1)
      ((new_C_mfsm_state = CMA1) \/
       (new_C_mfsm_state = CMA0) \/
       (new_C_mfsm_state = CMA2) \/
       (new_C_mfsm_state = CMD1) \/
```

```
          ((new_C_mfsm_state = CMD0) /\ C_mfsm_last_) \/
          (new_C_mfsm_state = CMW) \/
          (new_C_mfsm_state = CMABT))) in
    let ms210 =
       (ALTER
          ms10
          (2)
          ((new_C_mfsm_state = CMA3) \/
           (new_C_mfsm_state = CMA1) \/
           (new_C_mfsm_state = CMA0) \/
           (new_C_mfsm_state = CMA2) \/
           (new_C_mfsm_state = CMD1) \/
           (new_C_mfsm_state = CMD0) \/
           (new_C_mfsm_state = CMW))) in
    let mfsm_ms = ms210 in
    let mfsm_rqt_ = (new_C_mfsm_state = CMI) in
    let mfsm_cgnt_ = (~(new_C_mfsm_state = CMA3)) in
    let mfsm_cm_en = (~(new_C_mfsm_state = CMI) /\ ~(new_C_mfsm_state = CMR)) in
    let mfsm_abort_le_en_ =
       (~(new_C_mfsm_state = CMABT) \/ (new_C_mfsm_state = CMI)) in
%ok to here afterwards death due to frame stack overflow%
    let mfsm_mparity = ((new_C_mfsm_state = CMA3) \/
                        (new_C_mfsm_state = CMA1) \/
                        (new_C_mfsm_state = CMA0) \/
                        (new_C_mfsm_state = CMA2) \/
                        (new_C_mfsm_state = CMD1) \/
                        (new_C_mfsm_state = CMD0) \/
                        (C_mfsm_state = CMA1) \/
                        (C_mfsm_state = CMA0) \/
                        (C_mfsm_state = CMA2) \/
                        (C_mfsm_state = CMD1)) in
    let new_C_sfsm_state =
       ((C_sfsm_rst) => CSI |
        (C_sfsm_state = CSI) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MSTART) /\ ~C_sfsm_grant /\
             C_sfsm_addressed)
               => CSA1 | CSI) |
        (C_sfsm_state = CSL) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MSTART) /\ ~C_sfsm_grant /\
             C_sfsm_addressed)
               => CSA1 |
            (C_sfsm_D /\ (C_sfsm_ms = ^MSTART) /\ ~C_sfsm_grant /\
             ~C_sfsm_addressed)
               => CSI |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSL) |
        (C_sfsm_state = CSA1) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSA0 |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSA1) |
        (C_sfsm_state = CSA0) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY) /\ ~C_sfsm_hlda_) => CSALE |
            (C_sfsm_D /\ (C_sfsm_ms = ^MRDY) /\ C_sfsm_hlda_) => CSA0W |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSA0) |
        (C_sfsm_state = CSA0W) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY) /\ ~C_sfsm_hlda_) => CSALE |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSA0W) |
        (C_sfsm_state = CSALE) =>
           ((C_sfsm_D /\ C_sfsm_write /\ (C_sfsm_ms = ^MRDY)) => CSD1 |
            (C_sfsm_D /\ ~C_sfsm_write /\ (C_sfsm_ms = ^MRDY)) => CSRR |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSALE) |
        (C_sfsm_state = CSRR) =>
           ((C_sfsm_D /\ ~(C_sfsm_ms = ^MABORT)) => CSD1 |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSRR) |
        (C_sfsm_state = CSD1) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSD0 |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSD1) |
        (C_sfsm_state = CSD0) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MEND)) => CSACK |
            (C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSD1 |
            (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSD0) |
        (C_sfsm_state = CSACK) =>
           ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSL |
            (C_sfsm_D /\ (C_sfsm_ms = ^MWAIT)) => CSI |
```

```
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSACK) |
    (C_sfsm_D) => CSI | CSABT) in
let ss0 =
   (ALTER
      ARBN
      (0)
      ((new_C_sfsm_state = CSA0W) \/
       ((new_C_sfsm_state = CSALE) /\ ~C_sfsm_write) \/
       (new_C_sfsm_state = CSACK))) in
let ss10 =
   (ALTER
      ss0
      (1)
      (~(new_C_sfsm_state = CSI) /\
       ~(new_C_sfsm_state = CSACK) /\
       ~(new_C_sfsm_state = CSABT))) in
let ss210 =
   (ALTER
      ss10
      (2)
      (~(new_C_sfsm_state = CSI) /\ ~(new_C_sfsm_state = CSABT))) in
let sfsm_ss = ss210 in
let sfsm_iad_en_s =
   (((new_C_sfsm_state = CSALE) /\ ~(C_sfsm_state = CSALE)) \/
    ((new_C_sfsm_state = CSALE) /\ C_sfsm_write) \/
    ((new_C_sfsm_state = CSD1) /\ C_sfsm_write /\ ~(C_sfsm_state = CSRR)) \/
    ((new_C_sfsm_state = CSD0) /\ C_sfsm_write) /\
    ((new_C_sfsm_state = CSACK) /\ C_sfsm_write)) in
let sfsm_sidle = (new_C_sfsm_state = CSI) in
let sfsm_slock = (new_C_sfsm_state = CSL) in
let sfsm_sa1 = (new_C_sfsm_state = CSA1) in
let sfsm_sa0 = (new_C_sfsm_state = CSA0) in
let sfsm_sale = (new_C_sfsm_state = CSALE) in
let sfsm_sd1 = (new_C_sfsm_state = CSD1) in
let sfsm_sd0 = (new_C_sfsm_state = CSD0) in
let sfsm_sack = (new_C_sfsm_state = CSACK) in
let sfsm_sabort = (new_C_sfsm_state = CSABT) in
let sfsm_s_cout_sel0 = (new_C_sfsm_state = CSD1) in
let sfsm_sparity = (~(new_C_sfsm_state = CSI) /\
                    ~(new_C_sfsm_state = CSACK) /\
                    ~(new_C_sfsm_state = CSABT)) in
let new_C_efsm_state =
   ((C_efsm_rst) => CEI |
    (C_efsm_state = CEI) => ((~C_efsm_cale_) => CEE | CEI) |
    ((~C_efsm_last_ /\ ~C_efsm_srdy_) \/ ~C_efsm_male_ \/ ~C_efsm_rale_)
       => CEI | CEE) in
let efsm_srdy_en = ((new_C_efsm_state = CEE) \/ (C_efsm_state = CEE)) in
let new_C_lock_in_ =
   ((BSel(Rst) \/ mfsm_ma1)
      => ((BSel(Rst)) => F | BSel(I_lock_))
      | C_lock_in_) in
let new_C_last_in_ =
   ((ASel(Rst) \/ (ASel(ClkD) /\ mfsm_md1) \/ mfsm_mabort)
      => ((ASel(Rst)) => F | ASel(I_last_in_))
      | C_last_in_) in
let new_C_ss =
   ((mfsm_abort_le_en_) => ASel(CB_ss_in) | C_ss) in
let new_C_clkA = BSel(ClkD) in
let mend = ((ASel(CB_ms_in) = ^MEND), (BSel(CB_ms_in) = ^MEND)) in
let mabort = ((ASel(CB_ms_in) = ^MABORT), (BSel(CB_ms_in) = ^MABORT)) in
let last_out_inS = sfsm_sa1 in
let last_out_inR = (BSel(ClkD) /\ (BSel(mend) \/ BSel(mabort))) in
let new_C_last_out_ =
   ((last_out_inS \/ last_out_inR)
      => ((last_out_inS /\ ~last_out_inR) => T |
          (~last_out_inS /\ last_out_inR) => F |
          (~last_out_inS /\ ~last_out_inR) => F | ARB)
      | C_last_out_) in
let new_C_sidle_del = sfsm_sidle in
let new_C_mrqt_del = mfsm_mrequest in
let new_C_hold_ =
   ((BSel(ClkD)) => sfsm_sidle | C_hold_) in
```

```
let new_C_wr =
   ((~BSel(I_cale_)) => (ELEMENT (BSel(I_ad_in)) (27)) | C_wr) in
let mfsm_cin_4_leB = (C_clkA /\ sfsm_sa0) in
let new_C_sizewrbe =
   ((BSel(Rst)) => WORDN 9 0 |
    (mfsm_cin_4_leB) => (SUBARRAY C_data_in (31,22))
                          |  C_sizewrbe) in
let write =
   (((mfsm_cm_en) => C_wr | (ELEMENT C_sizewrbe (5))),
    ((mfsm_cm_en) => new_C_wr | (ELEMENT new_C_sizewrbe (5)))) in
let srdy = ((ASel(CB_ss_in) = ^SRDY), (BSel(CB_ss_in) = ^SRDY)) in
let dfsm_master =
   ((mfsm_ma3 \/ mfsm_ma2 \/ mfsm_ma1 \/ mfsm_ma0 \/ mfsm_md1 \/
     mfsm_md0),
    (mfsm_ma3 \/ mfsm_ma2 \/ mfsm_ma1 \/ mfsm_ma0 \/ mfsm_md1 \/
     mfsm_md0)) in
let dfsm_slave =
   ((~sfsm_sidle /\ ~sfsm_slock), (~sfsm_sidle /\ ~sfsm_slock)) in
let dfsm_cin_0_leB =
    (BSel(ClkD) /\
     ((mfsm_md0 /\ BSel(srdy) /\ ~BSel(write)) \/
      sfsm_sa0 \/
      (sfsm_sd0 /\ BSel(write)))) in
let dfsm_cin_1_leB = (BSel(ClkD) /\
                      ((mfsm_md1 /\ BSel(srdy) /\ ~BSel(write)) \/
                       sfsm_sa1 \/
                       (sfsm_sd1 /\ BSel(write)))) in
let dfsm_cin_3_leB = (BSel(ClkD) /\ (sfsm_sidle \/ sfsm_slock)) in
let dfsm_cout_0_leB =
    (BSel(I_cale_) \/
     (BSel(I_srdy_in_) /\ ~BSel(write)) \/
     (mfsm_ma0 /\ BSel(srdy) /\ BSel(write) /\ BSel(ClkD)) \/
     (mfsm_md0 /\ BSel(srdy) /\ BSel(write) /\ BSel(ClkD))) in
let dfsm_cout_1_leA = (C_clkA /\ sfsm_sd1) in
let dfsm_cad_en_ =
   ((~(mfsm_ma3 \/
       mfsm_ma2 \/
       mfsm_ma1 \/
       mfsm_ma0 \/
       (ASel(write) /\ (mfsm_md1 \/ mfsm_md0)) \/
       (~ASel(write) /\ (sfsm_sd1 \/ sfsm_sd0)))),
    (~(mfsm_ma3 \/
       mfsm_ma2 \/
       mfsm_ma1 \/
       mfsm_ma0 \/
       (BSel(write) /\ (mfsm_md1 \/ mfsm_md0)) \/
       (~BSel(write) /\ (sfsm_sd1 \/ sfsm_sd0))))) in
let dfsm_male_ =
 ((~(sfsm_sale /\ ~(VAL 1 (SUBARRAY C_sizewrbe (1,0)) = 3) /\ C_clkA)),
  (~(sfsm_sale /\ ~(VAL 1 (SUBARRAY new_C_sizewrbe (1,0)) = 3) /\ C_clkA)))
in
let dfsm_rale_ =
 ((~(sfsm_sale /\ (VAL 1 (SUBARRAY C_sizewrbe (1,0)) = 3) /\ C_clkA)),
  (~(sfsm_sale /\ (VAL 1 (SUBARRAY new_C_sizewrbe (1,0)) = 3) /\ C_clkA)))
in
let dfsm_mrdyB_ =
   (~((~BSel(write) /\ BSel(ClkD) /\ (sfsm_sale \/ sfsm_sd1)) \/
      (~BSel(write) /\ C_clkA /\ sfsm_sack) \/
      (BSel(write) /\ BSel(ClkD) /\ sfsm_sd0))) in
let new_C_cout_0_le_del = dfsm_cout_0_leB in
let new_C_cin_2_le = dfsm_cin_0_leB in
let new_C_mrdy_del_ = dfsm_mrdyB_ in
let new_C_iad_en_s_del = ((BSel(ClkD)) => sfsm_iad_en_s | C_iad_en_s_del) in
let new_C_wrdy = (BSel(srdy) /\ BSel(write) /\ mfsm_md1 /\ BSel(ClkD)) in
let new_C_rrdy = (BSel(srdy) /\ ~BSel(write) /\ mfsm_md0 /\ BSel(ClkD)) in
let pe_cntB = (BSel(ClkD) /\
               (~(sfsm_sparity = mfsm_mparity) \/
                ((SUBARRAY (BSel(CB_ss_in)) (1,0)) = WORDN 1 0))) in
let parity_inS =
      ((Par_Det rep (BSel(CB_ad_in))) /\ BSel(ClkD) /\ pe_cntB) in
let parity_inR = (BSel(Rst) \/ BSel(Reset_error)) in
let new_C_parity =
```

```
   ((parity_inS \/ parity_inR)
       => ((parity_inS /\ ~parity_inR) => T |
           (~parity_inS /\ parity_inR) => F |
           (~parity_inS /\ ~parity_inR) => F | ARB)
       |  C_parity) in
let new_C_source =
   ((BSel(Rst)) => WORDN 15 0 |
    (dfsm_cin_3_leB) => (Par_Dec rep (BSel(CB_ad_in))) | C_source) in
let di31_16 =
   (MALTER
      C_data_in
      (31,16)
      ((BSel(Rst)) => WORDN 15 0 |
       (dfsm_cin_1_leB)
          => (Par_Dec rep (BSel(CB_ad_in)))
          |  (SUBARRAY C_data_in (31,16)))) in
let di31_0 =
   (MALTER
      di31_16
      (15,0)
      ((BSel(Rst)) => WORDN 15 0 |
       (dfsm_cin_0_leB)
          => (Par_Dec rep (BSel(CB_ad_in)))
          |  (SUBARRAY C_data_in (15,0)))) in
let new_C_data_in = di31_0 in
let new_C_iad_out = (C_cin_2_le => C_data_in | C_iad_out) in
let cout_1_leA =
   ((dfsm_cout_1_leA /\ ~ASel(dfsm_master)) \/
    (ASel(dfsm_master) /\ C_cout_0_le_del)) in
let new_C_a1a0 = (cout_1_leA => C_iad_in | C_a1a0) in
let new_C_a3a2 = (mfsm_mrequest => (ASel(Ccr)) | C_a3a2) in
let new_C_iad_in = (dfsm_cout_0_leB => (BSel(I_ad_in)) | C_iad_in) in
let grantB =
   ((((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 0) /\
           ~(ELEMENT (BSel(CB_rqt_in_)) (0))   ) \/
    (((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 1) /\
           ~(ELEMENT (BSel(CB_rqt_in_)) (0)) /\
            (ELEMENT (BSel(CB_rqt_in_)) (1))   ) \/
    (((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 2) /\
           ~(ELEMENT (BSel(CB_rqt_in_)) (0)) /\
            (ELEMENT (BSel(CB_rqt_in_)) (1)) /\
            (ELEMENT (BSel(CB_rqt_in_)) (2))   ) \/
    (((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 3) /\
           ~(ELEMENT (BSel(CB_rqt_in_)) (0)) /\
            (ELEMENT (BSel(CB_rqt_in_)) (1)) /\
            (ELEMENT (BSel(CB_rqt_in_)) (2)) /\
            (ELEMENT (BSel(CB_rqt_in_)) (3))))) in
let busyB = (~((SUBARRAY (BSel(CB_rqt_in_)) (3,1)) = (WORDN 2 7))) in
let addressedB = (BSel(Id) = (SUBARRAY new_C_source (15,10))) in
let new_C_mfsm_srdy_en = sfsm_srdy_en in
let new_C_mfsm_D = (BSel(ClkD)) in
let new_C_mfsm_grant = grantB in
let new_C_mfsm_rst = (BSel(Rst)) in
let new_C_mfsm_busy = busyB in
let new_C_mfsm_write = (BSel(write)) in
let new_C_mfsm_crqt_ = (BSel(I_crqt_)) in
let new_C_mfsm_hold_ = C_hold_ in
let new_C_mfsm_last_ = new_C_last_in_ in
let new_C_mfsm_lock_ = new_C_lock_in_ in
let new_C_mfsm_ss = (BSel(CB_ss_in)) in
let new_C_mfsm_invalid = (BSel(Piu_invalid)) in
let new_C_sfsm_D = (BSel(ClkD)) in
let new_C_sfsm_grant = grantB in
let new_C_sfsm_rst = (BSel(Rst)) in
let new_C_sfsm_write = (BSel(write)) in
let new_C_sfsm_addressed = addressedB in
let new_C_sfsm_hlda_ = (BSel(I_hlda_)) in
let new_C_sfsm_ms = (BSel(CB_ms_in)) in
let new_C_efsm_cale_ = (BSel(I_cale_)) in
let new_C_efsm_last_ = (BSel(I_last_in_)) in
let new_C_efsm_male_ = (BSel(I_male_in_)) in
let new_C_efsm_rale_ = (BSel(I_rale_in_)) in
```

```
    let new_C_efsm_srdy_ = (BSel(I_srdy_in_)) in
    let new_C_efsm_rst = (BSel(Rst)) in

    (CCState new_C_mfsm_state new_C_mfsm_srdy_en new_C_mfsm_D new_C_mfsm_grant
             new_C_mfsm_rst new_C_mfsm_busy new_C_mfsm_write new_C_mfsm_crqt_
             new_C_mfsm_hold_ new_C_mfsm_last_ new_C_mfsm_lock_ new_C_mfsm_ss
             new_C_mfsm_invalid new_C_sfsm_state new_C_sfsm_D new_C_sfsm_grant
             new_C_sfsm_rst new_C_sfsm_write new_C_sfsm_addressed
             new_C_sfsm_hlda_ new_C_sfsm_ms new_C_efsm_state new_C_efsm_cale_
             new_C_efsm_last_ new_C_efsm_male_ new_C_efsm_rale_ new_C_efsm_srdy_
             new_C_efsm_rst new_C_lock_in_ new_C_last_in_ new_C_ss new_C_clkA
             new_C_last_out_ new_C_sidle_del new_C_mrqt_del new_C_hold_
             new_C_cout_0_le_del new_C_cin_2_le new_C_mrdy_del_
             new_C_iad_en_s_del new_C_wrdy new_C_rrdy new_C_parity new_C_source
             new_C_data_in new_C_sizewrbe new_C_iad_out new_C_a1a0 new_C_a3a2
             new_C_iad_in new_C_wr)"
  );;

let CC_NSF_REW = save_thm
   ('CC_NSF_REW',
    (PURE_ONCE_REWRITE_RULE [ASel;BSel] CC_NSF)
   );;

%-----------------------------------------------------------------------------
Output definition for C-Port instruction.
-----------------------------------------------------------------------------%

let CC_OF = new_definition
('CC_OF',
 "! (rep :^REP_ty) (s :cc_state) (e :cc_env) .
  CC_OF rep s e =
     let C_mfsm_state = C_mfsm_stateS s and
         C_mfsm_srdy_en = C_mfsm_srdy_enS s and
         C_mfsm_D = C_mfsm_DS s and
         C_mfsm_grant = C_mfsm_grantS s and
         C_mfsm_rst = C_mfsm_rstS s and
         C_mfsm_busy = C_mfsm_busyS s and
         C_mfsm_write = C_mfsm_writeS s and
         C_mfsm_crqt_ = C_mfsm_crqt_S s and
         C_mfsm_hold_ = C_mfsm_hold_S s and
         C_mfsm_last_ = C_mfsm_last_S s and
         C_mfsm_lock_ = C_mfsm_lock_S s and
         C_mfsm_ss = C_mfsm_ssS s and
         C_mfsm_invalid = C_mfsm_invalidS s and
         C_sfsm_state = C_sfsm_stateS s and
         C_sfsm_D = C_sfsm_DS s and
         C_sfsm_grant = C_sfsm_grantS s and
         C_sfsm_rst = C_sfsm_rstS s and
         C_sfsm_write = C_sfsm_writeS s and
         C_sfsm_addressed = C_sfsm_addressedS s and
         C_sfsm_hlda_ = C_sfsm_hlda_S s and
         C_sfsm_ms = C_sfsm_msS s and
         C_efsm_state = C_efsm_stateS s and
         C_efsm_cale_ = C_efsm_cale_S s and
         C_efsm_last_ = C_efsm_last_S s and
         C_efsm_male_ = C_efsm_male_S s and
         C_efsm_rale_ = C_efsm_rale_S s and
         C_efsm_srdy_ = C_efsm_srdy_S s and
         C_efsm_rst = C_efsm_rstS s and
         C_lock_in_ = C_lock_in_S s and
         C_last_in_ = C_last_in_S s and
         C_ss = C_ssS s and
         C_clkA = C_clkAS s and
         C_last_out_ = C_last_out_S s and
         C_sidle_del = C_sidle_delS s and
         C_mrqt_del = C_mrqt_delS s and
         C_hold_ = C_hold_S s and
         C_cout_0_le_del = C_cout_0_le_delS s and
         C_cin_2_le = C_cin_2_leS s and
         C_mrdy_del_ = C_mrdy_del_S s and
         C_iad_en_s_del = C_iad_en_s_delS s and
         C_wrdy = C_wrdyS s and
```

```
    C_rrdy = C_rrdyS s and
    C_parity = C_parityS s and
    C_source = C_sourceS s and
    C_data_in = C_data_inS s and
    C_sizewrbe = C_sizewrbeS s and
    C_iad_out = C_iad_outS s and
    C_a1a0 = C_a1a0S s and
    C_a3a2 = C_a3a2S s and
    C_iad_in = C_iad_inS s and
    C_wr = C_wrS s in
let I_ad_in = I_ad_inE e and
    I_be_in_ = I_be_in_E e and
    I_mrdy_in_ = I_mrdy_in_E e and
    I_rale_in_ = I_rale_in_E e and
    I_male_in_ = I_male_in_E e and
    I_last_in_ = I_last_in_E e and
    I_srdy_in_ = I_srdy_in_E e and
    I_lock_ = I_lock_E e and
    I_cale_ = I_cale_E e and
    I_hlda_ = I_hlda_E e and
    I_crqt_ = I_crqt_E e and
    CB_rqt_in_ = CB_rqt_in_E e and
    CB_ad_in = CB_ad_inE e and
    CB_ms_in = CB_ms_inE e and
    CB_ss_in = CB_ss_inE e and
    Rst = RstE e and
    ClkD = ClkDE e and
    Id = IdE e and
    ChannelID = ChannelIDE e and
    Pmm_failure = Pmm_failureE e and
    Piu_invalid = Piu_invalidE e and
    Ccr = CcrE e and
    Reset_error = Reset_errorE e in
let new_C_mfsm_state =
   ((C_mfsm_rst) => CMI |
    (C_mfsm_state = CMI) =>
       ((C_mfsm_D /\ ~C_mfsm_crqt_ /\ ~C_mfsm_busy /\ ~C_mfsm_invalid)
          => CMR | CMI) |
    (C_mfsm_state = CMR) =>
       ((C_mfsm_D /\ C_mfsm_grant /\ C_mfsm_hold_) => CMA3 | CMR) |
    (C_mfsm_state = CMA3) => ((C_mfsm_D) => CMA1 | CMA3) |
    (C_mfsm_state = CMA1) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMA0 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMA1) |
    (C_mfsm_state = CMA0) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMA2 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMA0) |
    (C_mfsm_state = CMA2) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMD1 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMA2) |
    (C_mfsm_state = CMD1) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY)) => CMD0 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMD1) |
    (C_mfsm_state = CMD0) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SRDY) /\ C_mfsm_last_) => CMD1 |
        (C_mfsm_D /\ (C_mfsm_ss = ^SRDY) /\ ~C_mfsm_last_) => CMW |
        (C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT | CMD0) |
    (C_mfsm_state = CMW) =>
       ((C_mfsm_D /\ (C_mfsm_ss = ^SABORT)) => CMABT |
        (C_mfsm_D /\ (C_mfsm_ss = ^SACK) /\ C_mfsm_lock_) => CMI |
        (C_mfsm_D /\ (C_mfsm_ss = ^SRDY) /\ ~C_mfsm_lock_ /\ ~C_mfsm_crqt_)
           => CMA3 | CMW) |
    (~C_mfsm_last_) => CMI | CMABT) in
let mfsm_mabort = (new_C_mfsm_state  = CMABT) in
let mfsm_midle = (new_C_mfsm_state = CMI) in
let mfsm_mrequest = (new_C_mfsm_state = CMR) in
let mfsm_ma3 = (new_C_mfsm_state = CMA3) in
let mfsm_ma2 = (new_C_mfsm_state = CMA2) in
let mfsm_ma1 = (new_C_mfsm_state = CMA1) in
let mfsm_ma0 = (new_C_mfsm_state = CMA0) in
let mfsm_md1 = (new_C_mfsm_state = CMD1) in
let mfsm_md0 = (new_C_mfsm_state = CMD0) in
```

```
let mfsm_iad_en_m =
   (((new_C_mfsm_state = CMD1) /\ ~C_mfsm_write /\ C_mfsm_srdy_en) \/
    ((new_C_mfsm_state = CMD0) /\ ~C_mfsm_write /\ C_mfsm_srdy_en) \/
    ((new_C_mfsm_state = CMW) /\ (C_mfsm_state = CMD0) /\ ~C_mfsm_write /\
     C_mfsm_srdy_en)) in
let mfsm_m_cout_sel1 =
   ((new_C_mfsm_state = CMA3) \/ (new_C_mfsm_state = CMA2)) in
let mfsm_m_cout_sel0 =
   ((new_C_mfsm_state = CMA3) \/ (new_C_mfsm_state = CMA1) \/
    (new_C_mfsm_state = CMD1)) in
let ms0 =
   (ALTER
      ARBN
      (0)
      (((new_C_mfsm_state = CMD0) /\ ~C_mfsm_last_) \/
       (new_C_mfsm_state = CMABT) \/
       ((new_C_mfsm_state = CMW) /\ C_mfsm_lock_))) in
let ms10 =
   (ALTER
      ms0
      (1)
      ((new_C_mfsm_state = CMA1) \/
       (new_C_mfsm_state = CMA0) \/
       (new_C_mfsm_state = CMA2) \/
       (new_C_mfsm_state = CMD1) \/
       ((new_C_mfsm_state = CMD0) /\ C_mfsm_last_) \/
       (new_C_mfsm_state = CMW) \/
       (new_C_mfsm_state = CMABT))) in
let ms210 =
   (ALTER
      ms10
      (2)
      ((new_C_mfsm_state = CMA3) \/
       (new_C_mfsm_state = CMA1) \/
       (new_C_mfsm_state = CMA0) \/
       (new_C_mfsm_state = CMA2) \/
       (new_C_mfsm_state = CMD1) \/
       (new_C_mfsm_state = CMD0) \/
       (new_C_mfsm_state = CMW))) in
let mfsm_ms = ms210 in
let mfsm_rqt_ = (new_C_mfsm_state = CMI) in
let mfsm_cgnt_ = (~(new_C_mfsm_state = CMA3)) in
let mfsm_cm_en = (~(new_C_mfsm_state = CMI) /\ ~(new_C_mfsm_state = CMR)) in
let mfsm_abort_le_en_ =
   (~(new_C_mfsm_state = CMABT) \/ (new_C_mfsm_state = CMI)) in
let mfsm_mparity = ((new_C_mfsm_state = CMA3) \/
                    (new_C_mfsm_state = CMA1) \/
                    (new_C_mfsm_state = CMA0) \/
                    (new_C_mfsm_state = CMA2) \/
                    (new_C_mfsm_state = CMD1) \/
                    (new_C_mfsm_state = CMD0) \/
                    (C_mfsm_state = CMA1) \/
                    (C_mfsm_state = CMA0) \/
                    (C_mfsm_state = CMA2) \/
                    (C_mfsm_state = CMD1)) in
let new_C_sfsm_state =
   ((C_sfsm_rst) => CSI |
    (C_sfsm_state = CSI) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MSTART) /\ ~C_sfsm_grant /\
         C_sfsm_addressed)
           => CSA1 | CSI) |
    (C_sfsm_state = CSL) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MSTART) /\ ~C_sfsm_grant /\
         C_sfsm_addressed)
           => CSA1 |
        (C_sfsm_D /\ (C_sfsm_ms = ^MSTART) /\ ~C_sfsm_grant /\
         ~C_sfsm_addressed)
           => CSI |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSL) |
    (C_sfsm_state = CSA1) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSA0 |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSA1) |
```

```
    (C_sfsm_state = CSA0) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY) /\ ~C_sfsm_hlda_) => CSALE |
        (C_sfsm_D /\ (C_sfsm_ms = ^MRDY) /\ C_sfsm_hlda_) => CSA0W |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSA0) |
    (C_sfsm_state = CSA0W) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY) /\ ~C_sfsm_hlda_) => CSALE |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSA0W) |
    (C_sfsm_state = CSALE) =>
       ((C_sfsm_D /\ C_sfsm_write /\ (C_sfsm_ms = ^MRDY)) => CSD1 |
        (C_sfsm_D /\ ~C_sfsm_write /\ (C_sfsm_ms = ^MRDY)) => CSRR |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSALE) |
    (C_sfsm_state = CSRR) =>
       ((C_sfsm_D /\ ~(C_sfsm_ms = ^MABORT)) => CSD1 |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSRR) |
    (C_sfsm_state = CSD1) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSD0 |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSD1) |
    (C_sfsm_state = CSD0) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MEND)) => CSACK |
        (C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSD1 |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSD0) |
    (C_sfsm_state = CSACK) =>
       ((C_sfsm_D /\ (C_sfsm_ms = ^MRDY)) => CSL |
        (C_sfsm_D /\ (C_sfsm_ms = ^MWAIT)) => CSI |
        (C_sfsm_D /\ (C_sfsm_ms = ^MABORT)) => CSABT | CSACK) |
    (C_sfsm_D) => CSI | CSABT) in
let ss0 =
   (ALTER
      ARBN
      (0)
      ((new_C_sfsm_state = CSA0W) \/
       ((new_C_sfsm_state = CSALE) /\ ~C_sfsm_write) \/
       (new_C_sfsm_state = CSACK))) in
let ss10 =
   (ALTER
      ss0
      (1)
      (~(new_C_sfsm_state = CSI) /\
       ~(new_C_sfsm_state = CSACK) /\
       ~(new_C_sfsm_state = CSABT))) in
let ss210 =
   (ALTER
      ss10
      (2)
      (~(new_C_sfsm_state = CSI) /\ ~(new_C_sfsm_state = CSABT))) in
let sfsm_ss = ss210 in
let sfsm_iad_en_s =
   (((new_C_sfsm_state = CSALE) /\ ~(C_sfsm_state = CSALE)) \/
    ((new_C_sfsm_state = CSALE) /\ C_sfsm_write) \/
    ((new_C_sfsm_state = CSD1) /\ C_sfsm_write /\ ~(C_sfsm_state = CSRR)) \/
    ((new_C_sfsm_state = CSD0) /\ C_sfsm_write) /\
    ((new_C_sfsm_state = CSACK) /\ C_sfsm_write)) in
let sfsm_sidle = (new_C_sfsm_state = CSI) in
let sfsm_slock = (new_C_sfsm_state = CSL) in
let sfsm_sa1 = (new_C_sfsm_state = CSA1) in
let sfsm_sa0 = (new_C_sfsm_state = CSA0) in
let sfsm_sale = (new_C_sfsm_state = CSALE) in
let sfsm_sd1 = (new_C_sfsm_state = CSD1) in
let sfsm_sd0 = (new_C_sfsm_state = CSD0) in
let sfsm_sack = (new_C_sfsm_state = CSACK) in
let sfsm_sabort = (new_C_sfsm_state = CSABT) in
let sfsm_s_cout_sel0 = (new_C_sfsm_state = CSD1) in
let sfsm_sparity = (~(new_C_sfsm_state = CSI) /\
                    ~(new_C_sfsm_state = CSACK) /\
                    ~(new_C_sfsm_state = CSABT)) in
let new_C_efsm_state =
   ((C_efsm_rst) => CEI |
    (C_efsm_state = CEI) => ((~C_efsm_cale_) => CEE | CEI) |
    ((~C_efsm_last_ /\ ~C_efsm_srdy_) \/ ~C_efsm_male_ \/ ~C_efsm_rale_)
       => CEI | CEE) in
let efsm_srdy_en = ((new_C_efsm_state = CEE) \/ (C_efsm_state = CEE)) in
let new_C_lock_in_ =
```

```
   ((BSel(Rst) \/ mfsm_ma1)
      => ((BSel(Rst)) => F | BSel(I_lock_))
      |  C_lock_in_) in
let new_C_last_in_ =
   ((ASel(Rst) \/ (ASel(ClkD) /\ mfsm_md1) \/ mfsm_mabort)
      => ((ASel(Rst)) => F | ASel(I_last_in_))
      |  C_last_in_) in
let new_C_ss =
   ((mfsm_abort_le_en_) => ASel(CB_ss_in) | C_ss) in
let new_C_clkA = BSel(ClkD) in
let mend = ((ASel(CB_ms_in) = ^MEND), (BSel(CB_ms_in) = ^MEND)) in
let mabort = ((ASel(CB_ms_in) = ^MABORT), (BSel(CB_ms_in) = ^MABORT)) in
let last_out_inS = sfsm_sa1 in
let last_out_inR = (BSel(ClkD) /\ (BSel(mend) \/ BSel(mabort))) in
let new_C_last_out_ =
   ((last_out_inS \/ last_out_inR)
      => ((last_out_inS /\ ~last_out_inR) => T |
          (~last_out_inS /\ last_out_inR) => F |
          (~last_out_inS /\ ~last_out_inR) => F | ARB)
      |  C_last_out_) in
let new_C_sidle_del = sfsm_sidle in
let new_C_mrqt_del = mfsm_mrequest in
let new_C_hold_ =
   ((BSel(ClkD)) => sfsm_sidle | C_hold_) in
let new_C_wr =
   ((~BSel(I_cale_)) => (ELEMENT (BSel(I_ad_in)) (27)) | C_wr) in
let mfsm_cin_4_leB = (C_clkA /\ sfsm_sa0) in
let new_C_sizewrbe =
   ((BSel(Rst)) => WORDN 9 0 |
    (mfsm_cin_4_leB) => (SUBARRAY C_data_in (31,22))
                        |  C_sizewrbe) in
let write =
   (((mfsm_cm_en) => C_wr | (ELEMENT C_sizewrbe (5))),
    ((mfsm_cm_en) => new_C_wr | (ELEMENT new_C_sizewrbe (5)))) in
let srdy = ((ASel(CB_ss_in) = ^SRDY), (BSel(CB_ss_in) = ^SRDY)) in
let dfsm_master =
   ((mfsm_ma3 \/ mfsm_ma2 \/ mfsm_ma1 \/ mfsm_ma0 \/ mfsm_md1 \/
     mfsm_md0),
    (mfsm_ma3 \/ mfsm_ma2 \/ mfsm_ma1 \/ mfsm_ma0 \/ mfsm_md1 \/
     mfsm_md0)) in
let dfsm_slave =
   ((~sfsm_sidle /\ ~sfsm_slock), (~sfsm_sidle /\ ~sfsm_slock)) in
let dfsm_cin_0_leB =
    (BSel(ClkD) /\
     ((mfsm_md0 /\ BSel(srdy) /\ ~BSel(write)) \/
      sfsm_sa0 \/
      (sfsm_sd0 /\ BSel(write)))) in
let dfsm_cin_1_leB = (BSel(ClkD) /\
                      ((mfsm_md1 /\ BSel(srdy) /\ ~BSel(write)) \/
                       sfsm_sa1 \/
                       (sfsm_sd1 /\ BSel(write)))) in
let dfsm_cin_3_leB = (BSel(ClkD) /\ (sfsm_sidle \/ sfsm_slock)) in
let dfsm_cout_0_leB =
    (BSel(I_cale_) \/
     (BSel(I_srdy_in_) /\ ~BSel(write)) \/
     (mfsm_ma0 /\ BSel(srdy) /\ BSel(write) /\ BSel(ClkD)) \/
     (mfsm_md0 /\ BSel(srdy) /\ BSel(write) /\ BSel(ClkD))) in
let dfsm_cout_1_leA = (C_clkA /\ sfsm_sd1) in
let dfsm_cad_en_ =
   ((~(mfsm_ma3 \/
       mfsm_ma2 \/
       mfsm_ma1 \/
       mfsm_ma0 \/
       (ASel(write) /\ (mfsm_md1 \/ mfsm_md0)) \/
       (~ASel(write) /\ (sfsm_sd1 \/ sfsm_sd0)))),
    (~(mfsm_ma3 \/
       mfsm_ma2 \/
       mfsm_ma1 \/
       mfsm_ma0 \/
       (BSel(write) /\ (mfsm_md1 \/ mfsm_md0)) \/
       (~BSel(write) /\ (sfsm_sd1 \/ sfsm_sd0))))) in
let dfsm_male_ =
```

```
 ((~(sfsm_sale /\ ~(VAL 1 (SUBARRAY C_sizewrbe (1,0)) = 3) /\ C_clkA)),
  (~(sfsm_sale /\ ~(VAL 1 (SUBARRAY new_C_sizewrbe (1,0)) = 3) /\ C_clkA)))
in
let dfsm_rale_ =
 ((~(sfsm_sale /\ (VAL 1 (SUBARRAY C_sizewrbe (1,0)) = 3) /\ C_clkA)),
  (~(sfsm_sale /\ (VAL 1 (SUBARRAY new_C_sizewrbe (1,0)) = 3) /\ C_clkA)))
in
let dfsm_mrdyB_ =
   (~((~BSel(write) /\ BSel(ClkD) /\ (sfsm_sale \/ sfsm_sd1)) \/
      (~BSel(write) /\ C_clkA /\ sfsm_sack) \/
      (BSel(write) /\ BSel(ClkD) /\ sfsm_sd0))) in
let new_C_cout_0_le_del = dfsm_cout_0_leB in
let new_C_cin_2_le = dfsm_cin_0_leB in
let new_C_mrdy_del_ = dfsm_mrdyB_ in
let new_C_iad_en_s_del = ((BSel(ClkD)) => sfsm_iad_en_s | C_iad_en_s_del) in
let new_C_wrdy = (BSel(srdy) /\ BSel(write) /\ mfsm_md1 /\ BSel(ClkD)) in
let new_C_rrdy = (BSel(srdy) /\ ~BSel(write) /\ mfsm_md0 /\ BSel(ClkD)) in
let pe_cntB = (BSel(ClkD) /\
               (~(sfsm_sparity = mfsm_mparity) \/
                ((SUBARRAY (BSel(CB_ss_in)) (1,0)) = WORDN 1 0))) in
let parity_inS =
      ((Par_Det rep (BSel(CB_ad_in))) /\ BSel(ClkD) /\ pe_cntB) in
let parity_inR = (BSel(Rst) \/ BSel(Reset_error)) in
let new_C_parity =
   ((parity_inS \/ parity_inR)
       => ((parity_inS /\ ~parity_inR) => T |
           (~parity_inS /\ parity_inR) => F |
           (~parity_inS /\ ~parity_inR) => F | ARB)
      |  C_parity) in
let new_C_source =
   ((BSel(Rst)) => WORDN 15 0 |
    (dfsm_cin_3_leB) => (Par_Dec rep (BSel(CB_ad_in))) | C_source) in
let di31_16 =
   (MALTER
      C_data_in
      (31,16)
      ((BSel(Rst)) => WORDN 15 0 |
       (dfsm_cin_1_leB)
          => (Par_Dec rep (BSel(CB_ad_in)))
          |  (SUBARRAY C_data_in (31,16)))) in
let di31_0 =
   (MALTER
      di31_16
      (15,0)
      ((BSel(Rst)) => WORDN 15 0 |
       (dfsm_cin_0_leB)
          => (Par_Dec rep (BSel(CB_ad_in)))
          |  (SUBARRAY C_data_in (15,0)))) in
let new_C_data_in = di31_0 in
let new_C_iad_out = (C_cin_2_le => C_data_in | C_iad_out) in
let cout_1_leA =
   ((dfsm_cout_1_leA /\ ~ASel(dfsm_master)) \/
    (ASel(dfsm_master) /\ C_cout_0_le_del)) in
let new_C_a1a0 = (cout_1_leA => C_iad_in | C_a1a0) in
let new_C_a3a2 = (mfsm_mrequest => (ASel(Ccr)) | C_a3a2) in
let new_C_iad_in = (dfsm_cout_0_leB => (BSel(I_ad_in)) | C_iad_in) in
let grantB =
   ((((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 0) /\
          ~(ELEMENT (BSel(CB_rqt_in_)) (0))   ) \/
    (((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 1) /\
          ~(ELEMENT (BSel(CB_rqt_in_)) (0)) /\
           (ELEMENT (BSel(CB_rqt_in_)) (1))   ) \/
    (((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 2) /\
          ~(ELEMENT (BSel(CB_rqt_in_)) (0)) /\
           (ELEMENT (BSel(CB_rqt_in_)) (1)) /\
           (ELEMENT (BSel(CB_rqt_in_)) (2))   ) \/
    (((SUBARRAY (BSel(Id)) (1,0)) = WORDN 1 3) /\
          ~(ELEMENT (BSel(CB_rqt_in_)) (0)) /\
           (ELEMENT (BSel(CB_rqt_in_)) (1)) /\
           (ELEMENT (BSel(CB_rqt_in_)) (2)) /\
           (ELEMENT (BSel(CB_rqt_in_)) (3))))) in
let busyB = (~((SUBARRAY (BSel(CB_rqt_in_)) (3,1)) = (WORDN 2 7))) in
```

```
let addressedB = (BSel(Id) = (SUBARRAY new_C_source (15,10))) in
let new_C_mfsm_srdy_en = efsm_srdy_en in
let new_C_mfsm_D = (BSel(ClkD)) in
let new_C_mfsm_grant = grantB in
let new_C_mfsm_rst = (BSel(Rst)) in
let new_C_mfsm_busy = busyB in
let new_C_mfsm_write = (BSel(write)) in
let new_C_mfsm_crqt_ = (BSel(I_crqt_)) in
let new_C_mfsm_hold_ = C_hold_ in
let new_C_mfsm_last_ = new_C_last_in_ in
let new_C_mfsm_lock_ = new_C_lock_in_ in
let new_C_mfsm_ss = (BSel(CB_ss_in)) in
let new_C_mfsm_invalid = (BSel(Piu_invalid)) in
let new_C_sfsm_D = (BSel(ClkD)) in
let new_C_sfsm_grant = grantB in
let new_C_sfsm_rst = (BSel(Rst)) in
let new_C_sfsm_write = (BSel(write)) in
let new_C_sfsm_addressed = addressedB in
let new_C_sfsm_hlda_ = (BSel(I_hlda_)) in
let new_C_sfsm_ms = (BSel(CB_ms_in)) in
let new_C_efsm_cale_ = (BSel(I_cale_)) in
let new_C_efsm_last_ = (BSel(I_last_in_)) in
let new_C_efsm_male_ = (BSel(I_male_in_)) in
let new_C_efsm_rale_ = (BSel(I_rale_in_)) in
let new_C_efsm_srdy_ = (BSel(I_srdy_in_)) in
let new_C_efsm_rst = (BSel(Rst)) in

let I_cgnt_ = (mfsm_cgnt_, mfsm_cgnt_) in
let I_mrdy_out_ =
   (((~ASel(I_hlda_)) => (WIRE C_mrdy_del_) | Z),
    ((~BSel(I_hlda_)) => (WIRE C_mrdy_del_) | Z)) in
let I_hold_ = (C_hold_, C_hold_) in
let I_rale_out_ =
   (((~ASel(I_hlda_)) => WIRE (ASel(dfsm_rale_)) | Z),
    ((~BSel(I_hlda_)) => WIRE (BSel(dfsm_rale_)) | Z)) in
let I_male_out_ =
   (((~ASel(I_hlda_)) => WIRE (ASel(dfsm_male_)) | Z),
    ((~BSel(I_hlda_)) => WIRE (BSel(dfsm_male_)) | Z)) in
let I_last_out_ =
   (((~ASel(I_hlda_)) => (WIRE C_last_out_) | Z),
    ((~BSel(I_hlda_)) => (WIRE new_C_last_out_) | Z)) in
let I_srdy_out_ =
   (((~ASel(I_cale_) \/ efsm_srdy_en) =>
        (WIRE ~(C_wrdy \/ C_rrdy \/ mfsm_mabort)) | Z),
    ((~BSel(I_cale_) \/ efsm_srdy_en) =>
        (WIRE ~(C_wrdy \/ C_rrdy \/ mfsm_mabort)) | Z)) in
let iad_en = (mfsm_iad_en_m \/ sfsm_iad_en_s \/ C_iad_en_s_del) in
let I_ad_out = ((iad_en => (BUSN new_C_iad_out) | Offn),
                (iad_en => (BUSN new_C_iad_out) | Offn)) in
let I_be_out_ =
   (((~ASel(I_hlda_)) => (BUSN (SUBARRAY C_sizewrbe (9,6))) | Offn),
    ((~BSel(I_hlda_)) => (BUSN (SUBARRAY new_C_sizewrbe (9,6))) | Offn)) in
let CB_rqt_out_ = (mfsm_rqt_, mfsm_rqt_) in
let ma1_0 = (MALTER
               ARBN
               (1,0)
               (SUBARRAY mfsm_ms (1,0))) in
let ma2_0 = (ALTER
               ma1_0
               (2)
               ((ELEMENT mfsm_ms (2)) /\
                ~ASel(Pmm_failure) /\ ~ASel(Piu_invalid))) in
let mb1_0 = (MALTER
               ARBN
               (1,0)
               (SUBARRAY mfsm_ms (1,0))) in
let mb2_0 = (ALTER
               mb1_0
               (2)
               ((ELEMENT mfsm_ms (2)) /\
                ~BSel(Pmm_failure) /\ ~BSel(Piu_invalid))) in
let CB_ms_out = (ma2_0, mb2_0) in
```

```
    let sa1_0 = (MALTER
                   ARBN
                   (1,0)
                   (SUBARRAY sfsm_ss (1,0))) in
    let sa2_0 = (ALTER
                   sa1_0
                   (2)
                   ((ELEMENT sfsm_ss (2)) /\
                    ~ASel(Pmm_failure) /\ ~ASel(Piu_invalid))) in
    let sb1_0 = (MALTER
                   ARBN
                   (1,0)
                   (SUBARRAY sfsm_ss (1,0))) in
    let sb2_0 = (ALTER
                   sb1_0
                   (2)
                   ((ELEMENT sfsm_ss (2)) /\
                    ~BSel(Pmm_failure) /\ ~BSel(Piu_invalid))) in
    let CB_ss_out = (sa2_0, sb2_0) in
    let cout_sel =
       ((sfsm_sd0 \/ sfsm_sd1) =>
           (let cs0 = (ALTER ARBN 0 sfsm_s_cout_sel0) in
            ALTER cs0 1 F) |
           (let cs0 = (ALTER ARBN 0 mfsm_m_cout_sel0) in
            ALTER cs0 1 (mfsm_m_cout_sel1))) in
    let CB_ad_out =
      (((~ASel(dfsm_cad_en_))
          => (BUSN
               ((cout_sel = WORDN 1 0) =>
                   (Par_Enc rep (SUBARRAY new_C_a1a0 (15,0))) |
                (cout_sel = WORDN 1 1) =>
                   (Par_Enc rep (SUBARRAY new_C_a1a0 (31,16))) |
                (cout_sel = WORDN 1 2) =>
                   (Par_Enc rep (SUBARRAY new_C_a3a2 (15,0)))
                |  (Par_Enc rep (SUBARRAY new_C_a3a2 (31,16)))))
        |  Offn),
       ((~BSel(dfsm_cad_en_))
          => (BUSN
               ((cout_sel = WORDN 1 0) =>
                   (Par_Enc rep (SUBARRAY new_C_a1a0 (15,0))) |
                (cout_sel = WORDN 1 1) =>
                   (Par_Enc rep (SUBARRAY new_C_a1a0 (31,16))) |
                (cout_sel = WORDN 1 2) =>
                   (Par_Enc rep (SUBARRAY new_C_a3a2 (15,0)))
                |  (Par_Enc rep (SUBARRAY new_C_a3a2 (31,16)))))
        |  Offn)) in
    let C_ss_out = (C_ss, new_C_ss) in
    let Disable_writes =
       ((ASel(dfsm_slave) /\
         ~(ELEMENT (SUBARRAY C_source (9,6)) (VAL 1 (ASel(ChannelID))))),
        (BSel(dfsm_slave) /\
         ~(ELEMENT (SUBARRAY new_C_source (9,6)) (VAL 1 (BSel(ChannelID)))))) in
    let CB_parity = (C_parity, new_C_parity) in

    (CCOut I_cgnt_ I_mrdy_out_ I_hold_ I_rale_out_ I_male_out_ I_last_out_
          I_srdy_out_ I_ad_out I_be_out_ CB_rqt_out_ CB_ms_out CB_ss_out
          CB_ad_out C_ss_out Disable_writes CB_parity)"
  );;

let CC_OF_REW = save_thm
   ('CC_OF_REW',
    (PURE_ONCE_REWRITE_RULE [ASel;BSel] CC_OF)
   );;

let CC_Exec = new_definition
   ('CC_Exec',
    "! (cci :CCI) (s :timeC->cc_state) (e :timeC->cc_env) (p :timeC->cc_out)
       (t :timeC) .
     CC_Exec cci s e p t = T"
   );;

let CC_PreC = new_definition
```

```
   ('CC_PreC',
    "! (cci :CCI) (s :timeC->cc_state) (e :timeC->cc_env) (p :timeC->cc_out)
       (t :timeC) .
     CC_PreC cci s e p t = T"
   );;

let CC_PostC = new_definition
   ('CC_PostC',
    "! (rep :^REP_ty) (cci :CCI) (s :timeC->cc_state) (e :timeC->cc_env)
       (p :timeC->cc_out) (t :timeC) .
     CC_PostC rep cci s e p t =
        (s (t+1) = CC_NSF rep (s t) (e t)) /\
        (p t = CC_OF rep (s t) (e t))"
   );;

let CC_Correct = new_definition
   ('CC_Correct',
    "! (rep :^REP_ty) (cci :CCI) (s :timeC->cc_state) (e :timeC->cc_env)
       (p :timeC->cc_out) (t :timeC) .
     CC_Correct rep cci s e p t =
        CC_Exec cci s e p t /\
        CC_PreC cci s e p t
        ==>
        CC_PostC rep cci s e p t"
   );;

 let CCSet_Correct = new_definition
   ('CCSet_Correct',
    "! (rep :^REP_ty) (s :timeC->cc_state) (e :timeC->cc_env) (p :timeC->cc_out).
     CCSet_Correct rep s e p = !(cci:CCI)(t:timeC). CC_Correct rep cci s e p t"
   );;

close_theory();;
```

3.6 SU-Cont Definitions

This section contains the theories *saux_def*, *sblock_def*, *sclock_def*, defining the SU-Cont design.

```
%-----------------------------------------------------------------------------

   File:        saux_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        1 March 1993

-----------------------------------------------------------------------------%

set_flag ('timing', true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftop/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

system 'rm saux_def.th';;

new_theory 'saux_def';;

loadf 'aux_defs';;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('wordn', ":(num->bool)");;

map load_parent ['piuaux_def'];;

%-----------------------------------------------------------------------------
  Abstract data type for the SU-Cont instruction.
```

```
------------------------------------------------------------------------------%

let SCI =
   define_type 'SCI'
               'SCI = SC_X';;

%-----------------------------------------------------------------------------
   Define abstract data type for the state.
------------------------------------------------------------------------------%

let s_state =
   define_type 's_state'
               's_state = SCState sfsm_ty bool bool bool bool bool bool bool
                                 wordn wordn bool bool bool bool bool bool bool
                                 bool bool bool';;

let S_fsm_stateS = new_recursive_definition
    false
    s_state
    'S_fsm_stateS'
    "S_fsm_stateS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                          S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                          S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                          S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                          S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_fsm_state";;

let S_fsm_rstS = new_recursive_definition
    false
    s_state
    'S_fsm_rstS'
    "S_fsm_rstS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                          S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                          S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                          S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                          S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_fsm_rst";;

let S_fsm_delay6S = new_recursive_definition
    false
    s_state
    'S_fsm_delay6S'
    "S_fsm_delay6S (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                          S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                          S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                          S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                          S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_fsm_delay6";;

let S_fsm_delay17S = new_recursive_definition
    false
    s_state
    'S_fsm_delay17S'
    "S_fsm_delay17S (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                          S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                          S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                          S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                          S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_fsm_delay17";;

let S_fsm_bothbadS = new_recursive_definition
    false
    s_state
    'S_fsm_bothbadS'
    "S_fsm_bothbadS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                          S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                          S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                          S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                          S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_fsm_bothbad";;

let S_fsm_bypassS = new_recursive_definition
```

```
    false
    s_state
    'S_fsm_bypassS'
    "S_fsm_bypassS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                    S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                    S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                    S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                    S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_fsm_bypass";;

let S_soft_shotS = new_recursive_definition
    false
    s_state
    'S_soft_shotS'
    "S_soft_shotS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                    S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                    S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                    S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                    S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_soft_shot";;

let S_soft_shot_delS = new_recursive_definition
    false
    s_state
    'S_soft_shot_delS'
    "S_soft_shot_delS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                    S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                    S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                    S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                    S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_soft_shot_del";;

let S_soft_cntS = new_recursive_definition
    false
    s_state
    'S_soft_cntS'
    "S_soft_cntS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                    S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                    S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                    S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                    S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_soft_cnt";;

let S_delayS = new_recursive_definition
    false
    s_state
    'S_delayS'
    "S_delayS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                    S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                    S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                    S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                    S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_delay";;

let S_instartS = new_recursive_definition
    false
    s_state
    'S_instartS'
    "S_instartS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                    S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                    S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                    S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                    S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_instart";;

let S_bad_cpu0S = new_recursive_definition
    false
    s_state
    'S_bad_cpu0S'
    "S_bad_cpu0S (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                    S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                    S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
```

```
                        S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                        S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_bad_cpu0";;

let S_bad_cpu1S = new_recursive_definition
    false
    s_state
    'S_bad_cpu1S'
    "S_bad_cpu1S (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                        S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                        S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                        S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                        S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_bad_cpu1";;

let S_reset_cpu0S = new_recursive_definition
    false
    s_state
    'S_reset_cpu0S'
    "S_reset_cpu0S (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                        S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                        S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                        S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                        S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_reset_cpu0";;

let S_reset_cpu1S = new_recursive_definition
    false
    s_state
    'S_reset_cpu1S'
    "S_reset_cpu1S (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                        S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                        S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                        S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                        S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_reset_cpu1";;

let S_cpu_bistS = new_recursive_definition
    false
    s_state
    'S_cpu_bistS'
    "S_cpu_bistS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                        S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                        S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                        S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                        S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_cpu_bist";;

let S_pmm_failS = new_recursive_definition
    false
    s_state
    'S_pmm_failS'
    "S_pmm_failS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                        S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                        S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                        S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                        S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_pmm_fail";;

let S_cpu0_failS = new_recursive_definition
    false
    s_state
    'S_cpu0_failS'
    "S_cpu0_failS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                        S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                        S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                        S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                        S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_cpu0_fail";;

let S_cpu1_failS = new_recursive_definition
    false
```

```
    s_state
    'S_cpu1_failS'
    "S_cpu1_failS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                          S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                          S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                          S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                          S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_cpu1_fail";;

let S_piu_failS = new_recursive_definition
    false
    s_state
    'S_piu_failS'
    "S_piu_failS (SCState S_fsm_state S_fsm_rst S_fsm_delay6 S_fsm_delay17
                          S_fsm_bothbad S_fsm_bypass S_soft_shot S_soft_shot_del
                          S_soft_cnt S_delay S_instart S_bad_cpu0 S_bad_cpu1
                          S_reset_cpu0 S_reset_cpu1 S_cpu_bist S_pmm_fail
                          S_cpu0_fail S_cpu1_fail S_piu_fail)
     = S_piu_fail";;

let State_CASES =
   prove_cases_thm (prove_induction_thm s_state);;

let State_Selectors_Work = prove_thm
   ('State_Selectors_Work',
    "! s:s_state .
     s = (SCState (S_fsm_stateS s) (S_fsm_rstS s) (S_fsm_delay6S s)
                  (S_fsm_delay17S s) (S_fsm_bothbadS s) (S_fsm_bypassS s)
                  (S_soft_shotS s) (S_soft_shot_delS s) (S_soft_cntS s)
                  (S_delayS s) (S_instartS s) (S_bad_cpu0S s) (S_bad_cpu1S s)
                  (S_reset_cpu0S s) (S_reset_cpu1S s) (S_cpu_bistS s)
                  (S_pmm_failS s) (S_cpu0_failS s) (S_cpu1_failS s)
                  (S_piu_failS s))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "s:s_state" State_CASES)
     THEN REWRITE_TAC [S_fsm_stateS; S_fsm_rstS; S_fsm_delay6S; S_fsm_delay17S;
                       S_fsm_bothbadS; S_fsm_bypassS; S_soft_shotS;
                       S_soft_shot_delS; S_soft_cntS; S_delayS; S_instartS;
                       S_bad_cpu0S; S_bad_cpu1S; S_reset_cpu0S; S_reset_cpu1S;
                       S_cpu_bistS; S_pmm_failS; S_cpu0_failS; S_cpu1_failS;
                       S_piu_failS]
   );;

%----------------------------------------------------------------------------
   Define abstract data type for the environment.
----------------------------------------------------------------------------%

let s_env =
   define_type 's_env'
               's_env = SCEnv bool#bool bool#bool bool#bool bool#bool bool#bool
                              bool#bool bool#bool';;

let RstE = new_recursive_definition
    false
    s_env
    'RstE'
    "RstE (SCEnv Rst Bypass Test Gcrh Gcrl Failure0_ Failure1_)
     = Rst";;

let BypassE = new_recursive_definition
    false
    s_env
    'BypassE'
    "BypassE (SCEnv Rst Bypass Test Gcrh Gcrl Failure0_ Failure1_)
     = Bypass";;

let TestE = new_recursive_definition
    false
    s_env
    'TestE'
    "TestE (SCEnv Rst Bypass Test Gcrh Gcrl Failure0_ Failure1_)
     = Test";;
```

```
let GcrhE = new_recursive_definition
    false
    s_env
    'GcrhE'
    "GcrhE (SCEnv Rst Bypass Test Gcrh Gcrl Failure0_ Failure1_)
     = Gcrh";;

let GcrlE = new_recursive_definition
    false
    s_env
    'GcrlE'
    "GcrlE (SCEnv Rst Bypass Test Gcrh Gcrl Failure0_ Failure1_)
     = Gcrl";;

let Failure0_E = new_recursive_definition
    false
    s_env
    'Failure0_E'
    "Failure0_E (SCEnv Rst Bypass Test Gcrh Gcrl Failure0_ Failure1_)
     = Failure0_";;

let Failure1_E = new_recursive_definition
    false
    s_env
    'Failure1_E'
    "Failure1_E (SCEnv Rst Bypass Test Gcrh Gcrl Failure0_ Failure1_)
     = Failure1_";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm s_env);;

let Env_Selectors_Work = prove_thm
   ('Env_Selectors_Work',
    "! e:s_env .
     e = (SCEnv (RstE e) (BypassE e) (TestE e) (GcrhE e) (GcrlE e) (Failure0_E e)
             (Failure1_E e))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "e:s_env" Env_CASES)
     THEN REWRITE_TAC [RstE; BypassE; TestE; GcrhE; GcrlE; Failure0_E;
                       Failure1_E]
   );;

%----------------------------------------------------------------------------
   Define abstract data type for the output.
----------------------------------------------------------------------------%

let s_out =
   define_type 's_out'
               's_out = SCOut wordn#wordn bool#bool bool#bool bool#bool bool#bool
                              bool#bool bool#bool bool#bool bool#bool bool#bool
                              bool#bool';;

let S_state0 = new_recursive_definition
    false
    s_out
    'S_state0'
    "S_state0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                     Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = S_state";;

let Reset_cport0 = new_recursive_definition
    false
    s_out
    'Reset_cport0'
    "Reset_cport0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Reset_cport";;

let Disable_int0 = new_recursive_definition
    false
    s_out
```

```
    'Disable_int0'
    "Disable_int0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Disable_int";;

let Reset_piu0 = new_recursive_definition
    false
    s_out
    'Reset_piu0'
    "Reset_piu0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Reset_piu";;

let Reset_cpu00 = new_recursive_definition
    false
    s_out
    'Reset_cpu00'
    "Reset_cpu00 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Reset_cpu0";;

let Reset_cpu10 = new_recursive_definition
    false
    s_out
    'Reset_cpu10'
    "Reset_cpu10 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Reset_cpu1";;

let Cpu_bist0 = new_recursive_definition
    false
    s_out
    'Cpu_bist0'
    "Cpu_bist0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Cpu_bist";;

let Piu_fail0 = new_recursive_definition
    false
    s_out
    'Piu_fail0'
    "Piu_fail0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Piu_fail";;

let Cpu0_fail0 = new_recursive_definition
    false
    s_out
    'Cpu0_fail0'
    "Cpu0_fail0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Cpu0_fail";;

let Cpu1_fail0 = new_recursive_definition
    false
    s_out
    'Cpu1_fail0'
    "Cpu1_fail0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Cpu1_fail";;

let Pmm_fail0 = new_recursive_definition
    false
    s_out
    'Pmm_fail0'
    "Pmm_fail0 (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0
                   Reset_cpu1 Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)
     = Pmm_fail";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm s_out);;
```

```
let Out_Selectors_Work = prove_thm
   ('Out_Selectors_Work',
    "! p:s_out .
     p = (SCOut (S_state0 p) (Reset_cport0 p) (Disable_int0 p) (Reset_piu0 p)
              (Reset_cpu00 p) (Reset_cpu10 p) (Cpu_bist0 p) (Piu_fail0 p)
              (Cpu0_fail0 p) (Cpu1_fail0 p) (Pmm_fail0 p))",
     GEN_TAC
     THEN STRUCT_CASES_TAC (SPEC "p:s_out" Out_CASES)
     THEN REWRITE_TAC [S_state0; Reset_cport0; Disable_int0; Reset_piu0;
                       Reset_cpu00; Reset_cpu10; Cpu_bist0; Piu_fail0;
                       Cpu0_fail0; Cpu1_fail0; Pmm_fail0]
   );;

close_theory();;


%----------------------------------------------------------------------------

   File:        sblock_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        4 March 1993

   This file contains the ml source for the gate-level specification of the
   startup controller of the FTEP PIU, an ASIC developed by the Embedded
   Processing Laboratory, Boeing High Technology Center.

----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/sucont/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/ml/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing', true);;

system 'rm sblock_def.th';;

new_theory 'sblock_def';;

loadf 'aux_defs';;

map new_parent ['saux_def';'array_def';'ineq'];;
map load_parent ['gates_def1';'latches_def';'ffs_def';'counters_def';
                 'piuaux_def';'wordn_def'];;

%----------------------------------------------------------------------------
Input logic for S_soft_shot latch.
----------------------------------------------------------------------------%

let Scnt_In_GATE = new_definition
   ('Scnt_In_GATE',
    "! (gcrh gcrl soft_shot_inD soft_cnt_inL :time->bool#bool) .
     Scnt_In_GATE gcrh gcrl soft_shot_inD soft_cnt_inL =
        ! t:time .
        (soft_shot_inD t = (((~ASel(gcrh t)) /\ ASel(gcrl t)),
                            ((~BSel(gcrh t)) /\ BSel(gcrl t)))) /\
        (soft_cnt_inL t = (((~ASel(gcrh t)) /\ ~ASel(gcrl t)),
                           ((~BSel(gcrh t)) /\ ~BSel(gcrl t))))"
   );;

%----------------------------------------------------------------------------
Input logic for S_soft_cnt counter.
----------------------------------------------------------------------------%

let Scnt_In1_GATE = new_definition
   ('Scnt_In1_GATE',
    "! (soft_shot_outQ soft_shot_del_outQ soft_cnt_inU :time->bool#bool) .
     Scnt_In1_GATE soft_shot_outQ soft_shot_del_outQ soft_cnt_inU =
        ! t:time .
```

```
        soft_cnt_inU t =
           ((ASel(soft_shot_outQ t) /\ (~ASel(soft_shot_del_outQ t))),
            (BSel(soft_shot_outQ t) /\ (~BSel(soft_shot_del_outQ t))))"
);;

%-----------------------------------------------------------------------------
Input logic for S_delay counter.
-----------------------------------------------------------------------------%

let Delay_In_GATE = new_definition
   ('Delay_In_GATE',
    "! (scpustart reset_cnt delay_inR :time->bool#bool)
       (delay :time->wordn#wordn) .
     Delay_In_GATE scpustart delay reset_cnt delay_inR =
        ! t:time .
        delay_inR t = ((ASel(reset_cnt t) \/
                        (ASel(scpustart t) /\ (ELEMENT (ASel(delay t)) (6)))),
                       (BSel(reset_cnt t) \/
                        (BSel(scpustart t) /\ (ELEMENT (BSel(delay t)) (6)))))"
);;

%-----------------------------------------------------------------------------
Delay counter output multiplexers.
-----------------------------------------------------------------------------%

let Muxes_GATE = new_definition
   ('Muxes_GATE',
    "! (test instart_inD delay17 :time->bool#bool)
       (delay :time->wordn#wordn) .
     Muxes_GATE delay test instart_inD delay17 =
        !t:time .
        (instart_inD t =
           (((ASel(test t)) => ELEMENT (ASel(delay t)) (5)
                             | ELEMENT (ASel(delay t)) (16)),
            ((BSel(test t)) => ELEMENT (BSel(delay t)) (5)
                             | ELEMENT (BSel(delay t)) (16)))) /\
        (delay17 t =
           (((ASel(test t)) => ELEMENT (ASel(delay t)) (6)
                             | ELEMENT (ASel(delay t)) (17)),
            ((BSel(test t)) => ELEMENT (BSel(delay t)) (6)
                             | ELEMENT (BSel(delay t)) (17))))"
   );;

%-----------------------------------------------------------------------------
Generation logic for Disable_int output.
-----------------------------------------------------------------------------%

let Dis_Int_Out_GATE = new_definition
   ('Dis_Int_Out_GATE',
    "! (instart normal disable_int_in disable_int_out :time->bool#bool)
       (delay :time->wordn#wordn) .
     Dis_Int_Out_GATE instart normal delay disable_int_in disable_int_out =
        ! t:time .
        (disable_int_out t =
            ((~ASel(instart t) /\
              (~ASel(normal t) \/ ~(ELEMENT (ASel(delay t)) (6))) /\
              ASel(disable_int_in t)),
             (~BSel(instart t) /\
              (~BSel(normal t) \/ ~(ELEMENT (BSel(delay t)) (6))) /\
              BSel(disable_int_in t))))"
   );;

%-----------------------------------------------------------------------------
Input logic for S_bad_cpu0, S_bad_cpu1 latches.
-----------------------------------------------------------------------------%

let Bad_Cpu_In_GATE = new_definition
   ('Bad_Cpu_In_GATE',
    "! (normal operation cpu0_fail cpu1_fail begin :time->bool#bool)
       (bad_cpu0_inS bad_cpu0_inR bad_cpu0_inE :time->bool#bool)
       (bad_cpu1_inS bad_cpu1_inR bad_cpu1_inE :time->bool#bool) .
     Bad_Cpu_In_GATE normal operation cpu0_fail cpu1_fail begin
```

```
                    bad_cpu0_inS bad_cpu0_inR bad_cpu0_inE
                    bad_cpu1_inS bad_cpu1_inR bad_cpu1_inE =
        ! t:time .
        let s_cpu0_select =
           (((ASel(normal t) \/ ASel(operation t)) /\ ~ASel(cpu0_fail t)),
            ((BSel(normal t) \/ BSel(operation t)) /\ ~BSel(cpu0_fail t))) in
        let s_cpu1_select =
           ((~ASel(cpu1_fail t) /\
             (ASel(normal t) \/ ASel(operation t)) /\
             ASel(cpu0_fail t)),
            (~BSel(cpu1_fail t) /\
             (BSel(normal t) \/ BSel(operation t)) /\
             BSel(cpu0_fail t))) in
        ((bad_cpu0_inS t = (ASel(begin t), BSel(begin t))) /\
         (bad_cpu0_inR t = s_cpu0_select) /\
         (bad_cpu0_inE t =
            ((ASel(s_cpu0_select) \/ ASel(begin t)),
             (BSel(s_cpu0_select) \/ BSel(begin t)))) /\
         (bad_cpu1_inS t = ((ASel(begin t)), (BSel(begin t)))) /\
         (bad_cpu1_inR t = s_cpu1_select) /\
         (bad_cpu1_inE t =
            ((ASel(s_cpu1_select) \/ ASel(begin t)),
             (BSel(s_cpu1_select) \/ BSel(begin t)))))"
   );;

%-----------------------------------------------------------------------------
Generation logic for local signals cpu0_ok, cpu1_ok.
-----------------------------------------------------------------------------%

let Cpu_Ok_GATE = new_definition
   ('Cpu_Ok_GATE',
    "! (cpu0_fail cpu1_fail failure0_ failure1_ cpu0_ok cpu1_ok :time->bool#bool)
       (soft_cnt :time->wordn#wordn) .
     Cpu_Ok_GATE soft_cnt cpu0_fail cpu1_fail failure0_ failure1_ cpu0_ok
                 cpu1_ok =
        ! t:time .
        (cpu0_ok t = ((ASel(cpu0_fail t) /\
                       ASel(failure0_ t) /\
                       ((ASel(soft_cnt t)) = WORDN 2 5)),
                      (BSel(cpu0_fail t) /\
                       BSel(failure0_ t) /\
                       ((BSel(soft_cnt t)) = WORDN 2 5)))) /\
        (cpu1_ok t = ((ASel(cpu1_fail t) /\
                       ASel(failure1_ t) /\
                       ((ASel(soft_cnt t)) = WORDN 2 5)),
                      (BSel(cpu1_fail t) /\
                       BSel(failure1_ t) /\
                       ((BSel(soft_cnt t)) = WORDN 2 5))))"
   );;

%-----------------------------------------------------------------------------
Input logic for S_pmm_fail, S_cpu0_fail, S_cpu1_fail, S_piu_fail latches.
-----------------------------------------------------------------------------%

let Fail_In_GATE = new_definition
   ('Fail_In_GATE',
    "! (begin pmm_fail piu_fail bypass cpu0_ok cpu1_ok :time->bool#bool)
       (pmm_fail_inS pmm_fail_inR pmm_fail_inE cpu0_fail_inS :time->bool#bool)
       (cpu0_fail_inR cpu0_fail_inE cpu1_fail_inS cpu1_fail_inR :time->bool#bool)
       (cpu1_fail_inE piu_fail_inS piu_fail_inR piu_fail_inE :time->bool#bool) .
     Fail_In_GATE begin pmm_fail piu_fail bypass cpu0_ok cpu1_ok
                  pmm_fail_inS pmm_fail_inR pmm_fail_inE cpu0_fail_inS
                  cpu0_fail_inR cpu0_fail_inE cpu1_fail_inS cpu1_fail_inR
                  cpu1_fail_inE piu_fail_inS piu_fail_inR piu_fail_inE =
        ! t:time .
        (pmm_fail_inS t = (ASel(begin t), BSel(begin t))) /\
        (pmm_fail_inR t = (ASel(pmm_fail t), BSel(pmm_fail t))) /\
        (pmm_fail_inE t = (((ASel(begin t)) \/ (ASel(pmm_fail t))),
                           ((BSel(begin t)) \/ (BSel(pmm_fail t))))) /\
        (cpu0_fail_inS t = (ASel(begin t), BSel(begin t))) /\
        (cpu0_fail_inR t = (((ASel(bypass t)) \/ (ASel(cpu0_ok t))),
                            ((BSel(bypass t)) \/ (BSel(cpu0_ok t))))) /\
```

```
        (cpu0_fail_inE t =
           (((ASel(begin t)) \/ (ASel(bypass t)) \/ (ASel(cpu0_ok t))),
            ((BSel(begin t)) \/ (BSel(bypass t)) \/ (BSel(cpu0_ok t))))) /\
        (cpu1_fail_inS t = (ASel(begin t), BSel(begin t))) /\
        (cpu1_fail_inR t = (((ASel(bypass t)) \/ (ASel(cpu1_ok t))),
                            ((BSel(bypass t)) \/ (BSel(cpu1_ok t))))) /\
        (cpu1_fail_inE t =
           (((ASel(begin t)) \/ (ASel(bypass t)) \/ (ASel(cpu1_ok t))),
            ((BSel(begin t)) \/ (BSel(bypass t)) \/ (BSel(cpu1_ok t))))) /\
        (piu_fail_inS t = (ASel(begin t), BSel(begin t))) /\
        (piu_fail_inR t = (((ASel(bypass t)) \/ (ASel(piu_fail t))),
                           ((BSel(bypass t)) \/ (BSel(piu_fail t))))) /\
        (piu_fail_inE t =
           (((ASel(begin t)) \/ (ASel(bypass t)) \/ (ASel(piu_fail t))),
            ((BSel(begin t)) \/ (BSel(bypass t)) \/ (BSel(piu_fail t)))))"
   );;

%-----------------------------------------------------------------------------
Startup-controller controller state machine.
-----------------------------------------------------------------------------%

let FSM_GATE = new_definition
   ('FSM_GATE',
    "! (rst_in delay17_in bothbad_in bypass_in :time->bool#bool)
       (delay_in :time->wordn#wordn)
       (rst delay6 delay17 bothbad bypass :time->bool)
       (state :time->sfsm_ty)
       (stateA_out :time->wordn#wordn)
       (sn_out so_out srcp_out sdi_out srp_out src0_out :time->bool#bool)
       (src1_out spf_out sc0f_out sc1f_out spmf_out sb_out :time->bool#bool)
       (src_out sec_out srs_out scs_out :time->bool#bool) .
     FSM_GATE rst_in delay_in delay17_in bothbad_in bypass_in
              state rst delay6 delay17 bothbad bypass
              stateA_out sn_out so_out srcp_out sdi_out srp_out src0_out
              src1_out spf_out sc0f_out sc1f_out spmf_out sb_out src_out
              sec_out srs_out scs_out =
       ! t:time .

       (state (t+1) =
          (rst t) => SSTART |
          ((state t) = SSTART) => SRA |
          ((state t) = SRA) => ((delay6 t) => ((bypass t) => SO | SPF) | SRA) |
          ((state t) = SPF) => SC0I |
          ((state t) = SC0I) => ((delay17 t) => SC0F | SC0I) |
          ((state t) = SC0F) => ST |
          ((state t) = ST) => SC1I |
          ((state t) = SC1I) => ((delay17 t) => SC1F | SC1I) |
          ((state t) = SC1F) => SS |
          ((state t)= SS) => ((bothbad t) => SSTOP | SCS) |
          ((state t) = SSTOP) => SSTOP |
          ((state t) = SCS) => ((delay6 t) => SN | SCS) |
          ((state t) = SN) => ((delay17 t) => SO | SN) | SO) /\
       (rst (t+1) = BSel(rst_in t)) /\
       (delay6 (t+1) = ELEMENT (BSel(delay_in t)) (6)) /\
       (delay17 (t+1) = BSel(delay17_in t)) /\
       (bothbad (t+1) = BSel(bothbad_in t)) /\
       (bypass (t+1) = BSel(bypass_in t)) /\

       (sn_out t = ((state (t+1) = SN), (state (t+1) = SN))) /\
       (so_out t = ((state (t+1) = SO), (state (t+1) = SO))) /\
       (let srcp = ((~(state (t+1) = SO) /\ ~(state t = SSTOP)) \/
                    (state t = SRA)) in
       (srcp_out t = (srcp, srcp))) /\
       (let sdi = ((~(state (t+1) = SO) /\ ~(state t = SSTOP)) \/
                   (state t = SRA)) in
       (sdi_out t = (sdi, sdi))) /\
       (let srp = ((state (t+1) = SSTART) \/ (state (t+1) = SRA) \/
                   (state (t+1) = SC0F) \/ (state (t+1) = ST) \/
                   (state (t+1) = SC1F) \/ (state (t+1) = SS) \/
                   (state (t+1) = SCS)) in
       (srp_out t = (srp, srp))) /\
       (let src0  = (~(state (t+1) = SPF) /\ ~(state (t+1) = SC0I)) in
```

```
        (src0_out t = (src0, src0))) /\
        (let src1 = (~(state (t+1) = ST) /\ ~(state (t+1) = SC1I)) in
        (src1_out t = (src1, src1))) /\
        (let spf = ((state t = SRA) /\ (delay6 t) /\ ~(rst t)) in
        (spf_out t = (spf, spf))) /\
        (sc0f_out t = ((state (t+1) = SC0F), (state (t+1) = SC0F))) /\
        (sc1f_out t = ((state (t+1) = SC1F), (state (t+1) = SC1F))) /\
        (spmf_out t = ((state (t+1) = SO), (state (t+1) = SO))) /\
        (sb_out t = ((state (t+1) = SSTART), (state (t+1) = SSTART))) /\
        (let src = ((state (t+1) = SSTART) \/
                    ((state t = SRA) /\ (delay6 t)) \/ (state (t+1) = SC0F) \/
                    (state (t+1) = ST) \/ (state (t+1) = SC1F) \/
                    (state (t+1) = SS) \/ ((state t = SCS) /\ delay6 t)) in
        (src_out t = (src, src))) /\
        (let sec = ((~(state (t+1) = SSTOP) /\ ~(state (t+1) = SO)) \/
                    (state t = SN)) in
        (sec_out t = (sec, sec))) /\
        (let srs = (((state t = SPF) /\ ~rst t) \/
                    ((state t = ST) /\ ~rst t)) in
        (srs_out t = (srs, srs))) /\
        (scs_out t = ((state (t+1) = SCS), (state (t+1) = SCS))) /\
        ((let a0 =
             (ALTER
                ARBN
                (0)
                ((state (t+1) = SRA) \/ (state (t+1) = SPF) \/
                 (state (t+1) = ST) \/ (state (t+1) = SC1I) \/
                 (state (t+1) = SCS) \/ (state (t+1) = SN) \/
                 (state (t+1) = SO))) in
         (let a1 =
             (ALTER
                a0
                (1)
                ((state (t+1) = SPF) \/ (state (t+1) = SC0I) \/
                 (state (t+1) = SC0F) \/ (state (t+1) = ST) \/
                 (state (t+1) = SSTOP) \/ (state (t+1) = SO))) in
         (let a2 =
             (ALTER
                a1
                (2)
                ((state (t+1) = SC0F) \/ (state (t+1) = ST) \/
                 (state (t+1) = SC1I) \/ (state (t+1) = SC1F) \/
                 (state (t+1) = SS) \/ (state (t+1) = SSTOP) \/
                 (state (t+1) = SCS))) in
         (let a3 =
             (ALTER
                a2
                (3)
                ((state (t+1) = SS) \/ (state (t+1) = SSTOP) \/
                 (state (t+1) = SCS) \/ (state (t+1) = SN) \/
                 (state (t+1) = SO))) in
         (stateA_out t = (a3, a3)))))))"
   );;

%-----------------------------------------------------------------------------
Startup controller block.
-----------------------------------------------------------------------------%

let SBlock_GATE = new_definition
   ('SBlock_GATE',
    "! (s :time->s_state) (e :time->s_env) (p :time->s_out) .
     SBlock_GATE s e p =
        ! t:time .
        ? (fsm_delay17 fsm_bothbad fsm_sn fsm_so fsm_sdi :time->bool#bool)
          (fsm_src0 fsm_src1 fsm_spf fsm_sc0f fsm_sc1f :time->bool#bool)
          (fsm_spmf fsm_sb fsm_src fsm_sec fsm_srs fsm_scs :time->bool#bool)
          (NC soft_shot_inD soft_shot_outQ soft_shot_del_outQ :time->bool#bool)
          (soft_cnt_inL soft_cnt_inU :time->bool#bool)
          (delay_inR instart_inD :time->bool#bool)
          (instart_outQ bad_cpu0_inS bad_cpu0_inR bad_cpu0_inE :time->bool#bool)
          (bad_cpu0_outQ reset_cpu0_inD bad_cpu1_inS :time->bool#bool)
          (bad_cpu1_inR bad_cpu1_inE bad_cpu1_outQ :time->bool#bool)
```

```
      (reset_cpu1_inD cpu_bist_inD cpu0_ok cpu1_ok :time->bool#bool)
      (pmm_fail_inS pmm_fail_inR pmm_fail_inE cpu0_fail_inS :time->bool#bool)
      (cpu0_fail_inR cpu0_fail_inE cpu1_fail_inS :time->bool#bool)
      (cpu1_fail_inR cpu1_fail_inE piu_fail_inS :time->bool#bool)
      (piu_fail_inR piu_fail_inE :time->bool#bool)
      (delay_outQ soft_cnt_outQ :time->wordn#wordn) .

    (Scnt_In_GATE (sig GcrhE e) (sig GcrlE e) soft_shot_inD soft_cnt_inL) /\
    (DLatA_GATE soft_shot_inD (sig S_soft_shotS s) soft_shot_outQ) /\
    (DFFA_GATE soft_shot_outQ (sig S_soft_shot_delS s) soft_shot_del_outQ) /\
    (Scnt_In1_GATE soft_shot_outQ soft_shot_del_outQ soft_cnt_inU) /\
    (UpRCntA_GATE 2 (GNDN 2) soft_cnt_inL soft_cnt_inU fsm_srs
                  (sig S_soft_cntS s) soft_cnt_outQ NC) /\
    (Delay_In_GATE fsm_scs delay_outQ fsm_src delay_inR) /\
    (UpRCntA_GATE 17 (GNDN 17) GND fsm_sec delay_inR (sig S_delayS s)
                  delay_outQ NC) /\
    (Muxes_GATE delay_outQ (sig TestE e) instart_inD fsm_delay17) /\
    (DLatA_GATE instart_inD (sig S_instartS s) instart_outQ) /\
    (Dis_Int_Out_GATE instart_outQ fsm_sn delay_outQ fsm_sdi
                      (sig Disable_intO p)) /\
    (AND2_GATE (sig Cpu0_failO p) (sig Cpu1_failO p) fsm_bothbad) /\
    (Bad_Cpu_In_GATE fsm_sn fsm_so (sig Cpu0_failO p) (sig Cpu1_failO p)
                     fsm_sb bad_cpu0_inS bad_cpu0_inR bad_cpu0_inE
                     bad_cpu1_inS bad_cpu1_inR bad_cpu1_inE) /\
    (DSRELatB_GATE GND bad_cpu0_inS bad_cpu0_inR bad_cpu0_inE
                   (sig S_bad_cpu0S s) bad_cpu0_outQ) /\
    (DSRELatB_GATE GND bad_cpu1_inS bad_cpu1_inR bad_cpu1_inE
                   (sig S_bad_cpu1S s) bad_cpu1_outQ) /\
    (AND2_GATE bad_cpu0_outQ fsm_src0 reset_cpu0_inD) /\
    (AND2_GATE bad_cpu1_outQ fsm_src1 reset_cpu1_inD) /\
    (DLatB_GATE reset_cpu0_inD (sig S_reset_cpu0S s) (sig Reset_cpu0O p)) /\
    (DLatB_GATE reset_cpu1_inD (sig S_reset_cpu1S s) (sig Reset_cpu1O p)) /\
    (AND3_GATE (sig Reset_cpu0O p) (sig Reset_cpu1O p) (sig BypassE e)
               cpu_bist_inD) /\
    (DFFB_GATE cpu_bist_inD (sig S_cpu_bistS s) (sig Cpu_bistO p)) /\
    (Fail_In_GATE fsm_sb fsm_spmf fsm_spf (sig BypassE e) cpu0_ok cpu1_ok
                  pmm_fail_inS pmm_fail_inR pmm_fail_inE cpu0_fail_inS
                  cpu0_fail_inR cpu0_fail_inE cpu1_fail_inS cpu1_fail_inR
                  cpu1_fail_inE piu_fail_inS piu_fail_inR piu_fail_inE) /\
    (DSRELatB_GATE GND pmm_fail_inS pmm_fail_inR pmm_fail_inE
                   (sig S_pmm_failS s) (sig Pmm_failO p)) /\
    (DSRELatB_GATE GND cpu0_fail_inS cpu0_fail_inR cpu0_fail_inE
                   (sig S_cpu0_failS s) (sig Cpu0_failO p)) /\
    (DSRELatB_GATE GND cpu1_fail_inS cpu1_fail_inR cpu1_fail_inE
                   (sig S_cpu1_failS s) (sig Cpu1_failO p)) /\
    (DSRELatB_GATE GND piu_fail_inS piu_fail_inR piu_fail_inE
                   (sig S_piu_failS s) (sig Piu_failO p)) /\
    (Cpu_Ok_GATE soft_cnt_outQ fsm_sc0f fsm_sc1f (sig Failure0_E e)
                 (sig Failure1_E e) cpu0_ok cpu1_ok) /\
    (FSM_GATE (sig RstE e) delay_outQ fsm_delay17 fsm_bothbad (sig BypassE e)
              (sig S_fsm_stateS s) (sig S_fsm_rstS s) (sig S_fsm_delay6S s)
              (sig S_fsm_delay17S s) (sig S_fsm_bothbadS s)
              (sig S_fsm_bypassS s) (sig S_stateO p) fsm_sn fsm_so
              (sig Reset_cportO p) fsm_sdi (sig Reset_piuO p) fsm_src0
              fsm_src1 fsm_spf fsm_sc0f fsm_sc1f fsm_spmf fsm_sb fsm_src
              fsm_sec fsm_srs fsm_scs)"
  );;

let SBlock = save_thm ('SBlock', SBlock_GATE);;

let SBlock_EXP = save_thm
   ('SBlock_EXP',
    (BETA_RULE
    (REWRITE_RULE [Scnt_In_GATE;Scnt_In1_GATE;Delay_In_GATE;Muxes_GATE;
                   Dis_Int_Out_GATE;EXPAND_LET_RULE(Bad_Cpu_In_GATE);
                   Fail_In_GATE;Cpu_Ok_GATE;EXPAND_LET_RULE(FSM_GATE);AND2_GATE;
                   AND3_GATE;DLatA_GATE;DLatB_GATE;DSRELatB_GATE;DFFA_GATE;
                   DFFB_GATE;UpRCntA_GATE;ASel;BSel;GND;GNDN;sig]
    (SPEC_ALL SBlock_GATE)))
  );;

close_theory();;
```

```
%-----------------------------------------------------------------------------

   File:        sclock_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        4 March 1993

This file contains the ml source for the clock-level specification of the startup
controller of the FTEP PIU, an ASIC developed by the Embedded Processing
Laboratory, Boeing High Technology Center.  The bulk of this code was translated
from an M-language simulation program using a translator written by P.J. Windley
at the University of Idaho.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/ftep/piu/hol/sucont/';
                                  '/home/elvis6/dfura/hol/ml/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm sclock_def.th';;

new_theory 'sclock_def';;

loadf 'aux_defs';;

map new_parent ['piuaux_def';'saux_def';'array_def';'wordn_def';'ineq'];;

new_type_abbrev ('timeC',":num");;

let ASel = definition 'piuaux_def' 'ASel';;
let BSel = definition 'piuaux_def' 'BSel';;
let RSTN = definition 'wordn_def' 'RSTN';;

%-----------------------------------------------------------------------------
Next-state definition for SU-Cont instruction.
-----------------------------------------------------------------------------%

let SC_NSF = new_definition
   ('SC_NSF',
    "! (s :s_state) (e :s_env) .
     SC_NSF s e =
        let S_fsm_state = S_fsm_stateS s and
            S_fsm_rst = S_fsm_rstS s and
            S_fsm_delay6 = S_fsm_delay6S s and
            S_fsm_delay17 = S_fsm_delay17S s and
            S_fsm_bothbad = S_fsm_bothbadS s and
            S_fsm_bypass = S_fsm_bypassS s and
            S_soft_shot = S_soft_shotS s and
            S_soft_shot_del = S_soft_shot_delS s and
            S_soft_cnt = S_soft_cntS s and
            S_delay = S_delayS s and
            S_instart = S_instartS s and
            S_bad_cpu0 = S_bad_cpu0S s and
            S_bad_cpu1 = S_bad_cpu1S s and
            S_reset_cpu0 = S_reset_cpu0S s and
            S_reset_cpu1 = S_reset_cpu1S s and
            S_cpu_bist = S_cpu_bistS s and
            S_pmm_fail = S_pmm_failS s and
            S_cpu0_fail = S_cpu0_failS s and
            S_cpu1_fail = S_cpu1_failS s and
            S_piu_fail = S_piu_failS s in
        let Rst = RstE e and
            Bypass = BypassE e and
            Test = TestE e and
            Gcrh = GcrhE e and
            Gcrl = GcrlE e and
```

```
    Failure0_ = Failure0_E e and
    Failure1_ = Failure1_E e in
let new_S_fsm_state =
   (S_fsm_rst => SSTART |
    (S_fsm_state = SSTART) => SRA |
    (S_fsm_state = SRA) =>
       (S_fsm_delay6 => (S_fsm_bypass => SO | SPF) | SRA) |
    (S_fsm_state = SPF) => SC0I |
    (S_fsm_state = SC0I) => (S_fsm_delay17 => SC0F | SC0I) |
    (S_fsm_state = SC0F) => ST |
    (S_fsm_state = ST) => SC1I |
    (S_fsm_state = SC1I) => (S_fsm_delay17 => SC1F | SC1I) |
    (S_fsm_state = SC1F) => SS |
    (S_fsm_state = SS) => (S_fsm_bothbad => SSTOP | SCS) |
    (S_fsm_state = SSTOP) => SSTOP |
    (S_fsm_state = SCS) => (S_fsm_delay6 => SN | SCS) |
    (S_fsm_state = SN) => (S_fsm_delay17 => SO | SN) | SO) in
let s_fsm_sn = (new_S_fsm_state = SN) in
let s_fsm_so = (new_S_fsm_state = SO) in
let s_fsm_srcp = (((~(new_S_fsm_state = SO)) /\ (~(S_fsm_state = SSTOP)))
                  \/ (S_fsm_state = SRA)) in
let s_fsm_sdi = (((~(new_S_fsm_state = SO)) /\ (~(S_fsm_state = SSTOP)))
                 \/ (S_fsm_state = SRA)) in
let s_fsm_srp = ((new_S_fsm_state = SSTART) \/ (new_S_fsm_state = SRA)
             \/ (new_S_fsm_state = SC0F) \/ (new_S_fsm_state = ST)
             \/ (new_S_fsm_state = SC1F) \/ (new_S_fsm_state = SS)
             \/ (new_S_fsm_state = SCS)) in
let s_fsm_src0 = ((~(new_S_fsm_state = SPF))
              /\ (~(new_S_fsm_state = SC0I))) in
let s_fsm_src1 = ((~(new_S_fsm_state = ST))
              /\ (~(new_S_fsm_state = SC1I))) in
let s_fsm_spf = ((S_fsm_state = SRA) /\ S_fsm_delay6 /\ ~S_fsm_rst) in
let s_fsm_sc0f = (new_S_fsm_state = SC0F) in
let s_fsm_sc1f = (new_S_fsm_state = SC1F) in
let s_fsm_spmf = (new_S_fsm_state = SO) in
let s_fsm_sb = (new_S_fsm_state = SSTART) in
let s_fsm_src = ((new_S_fsm_state = SSTART)
             \/ ((S_fsm_state = SRA) /\ S_fsm_delay6)
             \/ (new_S_fsm_state = SC0F) \/ (new_S_fsm_state = ST)
             \/ (new_S_fsm_state = SC1F) \/ (new_S_fsm_state = SS)
             \/ ((S_fsm_state = SCS) /\ S_fsm_delay6)) in
let s_fsm_sec = (((~(new_S_fsm_state = SSTOP)) /\
                  (~(new_S_fsm_state = SO))) \/ (S_fsm_state = SN)) in
let s_fsm_srs = (((S_fsm_state = SPF) /\ ~S_fsm_rst)
             \/ ((S_fsm_state = ST) /\ ~S_fsm_rst)) in
let s_fsm_scs = (new_S_fsm_state = SCS) in
let new_S_soft_shot = (~ASel(Gcrh) /\ ASel(Gcrl)) in
let new_S_soft_shot_del = new_S_soft_shot in
let s_soft_cnt_out =
   ((new_S_soft_shot_del /\ ~S_soft_shot_del)
     => (INCN 2 S_soft_cnt)
     |  S_soft_cnt) in
let new_S_soft_cnt =
   ((s_fsm_srs) => (WORDN 2 0) |
    (~BSel(Gcrh) /\ ~BSel(Gcrl)) => (RSTN 2) | s_soft_cnt_out) in
let s_delay_out =
   ((s_fsm_sec) => (INCN 17 S_delay) | S_delay) in
let new_S_delay =
  ((s_fsm_src \/ (s_fsm_scs /\ (ELEMENT (s_delay_out) (6))))
    => (WORDN 17 0)
    |  s_delay_out) in
let new_S_instart =
   ((ASel(Test))
    => (ELEMENT s_delay_out (5))
     | (ELEMENT s_delay_out (16))) in
let s_cpu0_ok =
   (s_fsm_sc0f /\ BSel(Failure0_) /\ (s_soft_cnt_out = (WORDN 2 5))) in
let s_cpu1_ok =
   (s_fsm_sc1f /\ BSel(Failure1_) /\ (s_soft_cnt_out = (WORDN 2 5))) in
let new_S_pmm_fail =
   ((s_fsm_sb \/ s_fsm_spmf)
    => ((s_fsm_sb /\ ~s_fsm_spmf) => T |
```

```
            (~s_fsm_sb /\ s_fsm_spmf) => F |
            (~s_fsm_sb /\ ~s_fsm_spmf) => F | ARB)
        |  S_pmm_fail) in
    let new_S_cpu0_fail =
       ((s_fsm_sb \/ BSel(Bypass) \/ s_cpu0_ok)
        => ((s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu0_ok)) => T |
            (~s_fsm_sb /\ (BSel(Bypass) \/ s_cpu0_ok)) => F |
            (~s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu0_ok)) => F | ARB)
        |  S_cpu0_fail) in
    let new_S_cpu1_fail =
       ((s_fsm_sb \/ BSel(Bypass) \/ s_cpu1_ok)
        => ((s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu1_ok)) => T |
            (~s_fsm_sb /\ (BSel(Bypass) \/ s_cpu1_ok)) => F |
            (~s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu1_ok)) => F | ARB)
        |  S_cpu1_fail) in
    let new_S_piu_fail =
       ((s_fsm_sb \/ BSel(Bypass) \/ s_fsm_spf)
        => ((s_fsm_sb /\ ~(BSel(Bypass) \/ s_fsm_spf)) => T |
            (~s_fsm_sb /\ (BSel(Bypass) \/ s_fsm_spf)) => F |
            (~s_fsm_sb /\ ~(BSel(Bypass) \/ s_fsm_spf)) => F | ARB)
        | S_piu_fail) in
    let s_cpu0_select = ((s_fsm_sn \/ s_fsm_so) /\ ~new_S_cpu0_fail) in
    let s_cpu1_select =
       (~new_S_cpu1_fail /\ (s_fsm_sn \/ s_fsm_so) /\ new_S_cpu0_fail) in
    let new_S_bad_cpu0 =
       ((s_cpu0_select \/ s_fsm_sb)
        => ((s_fsm_sb /\ ~s_cpu0_select) => T |
            (~s_fsm_sb /\ s_cpu0_select) => F |
            (~s_fsm_sb /\ ~s_cpu0_select) => F | ARB)
        |  S_bad_cpu0) in
    let new_S_bad_cpu1 =
       ((s_cpu1_select \/ s_fsm_sb)
        => ((s_fsm_sb /\ ~s_cpu1_select) => T |
            (~s_fsm_sb /\ s_cpu1_select) => F |
            (~s_fsm_sb /\ ~s_cpu1_select) => F | ARB)
        |  S_bad_cpu1) in
    let new_S_reset_cpu0 = (new_S_bad_cpu0 /\ s_fsm_src0) in
    let new_S_reset_cpu1 = (new_S_bad_cpu1 /\ s_fsm_src1) in
    let new_S_cpu_bist =
       (S_reset_cpu0 /\ S_reset_cpu1 /\ ASel(Bypass)) in
    let new_S_fsm_rst = BSel(Rst) in
    let new_S_fsm_delay6 = (ELEMENT s_delay_out (6)) in
    let new_S_fsm_delay17 =
       ((BSel(Test))
        => (ELEMENT s_delay_out (6))
         | (ELEMENT s_delay_out (17))) in
    let new_S_fsm_bothbad = (new_S_cpu0_fail /\ new_S_cpu1_fail) in
    let new_S_fsm_bypass = BSel(Bypass) in

    (SCState new_S_fsm_state new_S_fsm_rst new_S_fsm_delay6 new_S_fsm_delay17
           new_S_fsm_bothbad new_S_fsm_bypass new_S_soft_shot
           new_S_soft_shot_del new_S_soft_cnt new_S_delay new_S_instart
           new_S_bad_cpu0 new_S_bad_cpu1 new_S_reset_cpu0 new_S_reset_cpu1
           new_S_cpu_bist new_S_pmm_fail new_S_cpu0_fail new_S_cpu1_fail
           new_S_piu_fail)"
);;

let SClockNSF_REW = save_thm
   ('SClockNSF_REW',
    (REWRITE_RULE [ASel;BSel;RSTN] SC_NSF)
   );;

%-----------------------------------------------------------------------------
Output definition for SU-Cont instruction.
-----------------------------------------------------------------------------%

let SC_OF = new_definition
   ('SC_OF',
    "! (s :s_state) (e :s_env) .
     SC_OF s e =
        let S_fsm_state = S_fsm_stateS s and
            S_fsm_rst = S_fsm_rstS s and
```

```
    S_fsm_delay6 = S_fsm_delay6S s and
    S_fsm_delay17 = S_fsm_delay17S s and
    S_fsm_bothbad = S_fsm_bothbadS s and
    S_fsm_bypass = S_fsm_bypassS s and
    S_soft_shot = S_soft_shotS s and
    S_soft_shot_del = S_soft_shot_delS s and
    S_soft_cnt = S_soft_cntS s and
    S_delay = S_delayS s and
    S_instart = S_instartS s and
    S_bad_cpu0 = S_bad_cpu0S s and
    S_bad_cpu1 = S_bad_cpu1S s and
    S_reset_cpu0 = S_reset_cpu0S s and
    S_reset_cpu1 = S_reset_cpu1S s and
    S_cpu_bist = S_cpu_bistS s and
    S_pmm_fail = S_pmm_failS s and
    S_cpu0_fail = S_cpu0_failS s and
    S_cpu1_fail = S_cpu1_failS s and
    S_piu_fail = S_piu_failS s in
let Rst = RstE e and
    Bypass = BypassE e and
    Test = TestE e and
    Gcrh = GcrhE e and
    Gcrl = GcrlE e and
    Failure0_ = Failure0_E e and
    Failure1_ = Failure1_E e in
let new_S_fsm_state =
   (S_fsm_rst => SSTART |
    (S_fsm_state = SSTART) => SRA |
    (S_fsm_state = SRA) =>
       (S_fsm_delay6 => (S_fsm_bypass => SO | SPF) | SRA) |
    (S_fsm_state = SPF) => SC0I |
    (S_fsm_state = SC0I) => (S_fsm_delay17 => SC0F | SC0I) |
    (S_fsm_state = SC0F) => ST |
    (S_fsm_state = ST) => SC1I |
    (S_fsm_state = SC1I) => (S_fsm_delay17 => SC1F | SC1I) |
    (S_fsm_state = SC1F) => SS |
    (S_fsm_state = SS) => (S_fsm_bothbad => SSTOP | SCS) |
    (S_fsm_state = SSTOP) => SSTOP |
    (S_fsm_state = SCS) => (S_fsm_delay6 => SN | SCS) |
    (S_fsm_state = SN) => (S_fsm_delay17 => SO | SN) | SO) in
let s_fsm_sn = (new_S_fsm_state = SN) in
let s_fsm_so = (new_S_fsm_state = SO) in
let s_fsm_srcp = (((~(new_S_fsm_state = SO)) /\
                   (~(S_fsm_state = SSTOP)))
                    \/ (S_fsm_state = SRA)) in
let s_fsm_sdi = (((~(new_S_fsm_state = SO)) /\
                  (~(S_fsm_state = SSTOP)))
                   \/ (S_fsm_state = SRA)) in
let s_fsm_srp = ((new_S_fsm_state = SSTART) \/ (new_S_fsm_state = SRA)
              \/ (new_S_fsm_state = SC0F) \/ (new_S_fsm_state = ST)
              \/ (new_S_fsm_state = SC1F) \/ (new_S_fsm_state = SS)
              \/ (new_S_fsm_state = SCS)) in
let s_fsm_src0 = ((~(new_S_fsm_state = SPF))
               /\ (~(new_S_fsm_state = SC0I))) in
let s_fsm_src1 = ((~(new_S_fsm_state = ST))
               /\ (~(new_S_fsm_state = SC1I))) in
let s_fsm_spf = ((S_fsm_state = SRA) /\ S_fsm_delay6 /\ ~S_fsm_rst) in
let s_fsm_sc0f = (new_S_fsm_state = SC0F) in
let s_fsm_sc1f = (new_S_fsm_state = SC1F) in
let s_fsm_spmf = (new_S_fsm_state = SO) in
let s_fsm_sb = (new_S_fsm_state = SSTART) in
let s_fsm_src = ((new_S_fsm_state = SSTART)
              \/ ((S_fsm_state = SRA) /\ S_fsm_delay6)
              \/ (new_S_fsm_state = SC0F) \/ (new_S_fsm_state = ST)
              \/ (new_S_fsm_state = SC1F) \/ (new_S_fsm_state = SS)
              \/ ((S_fsm_state = SCS) /\ S_fsm_delay6)) in
let s_fsm_sec = (((~(new_S_fsm_state = SSTOP)) /\
                  (~(new_S_fsm_state = SO))) \/ (S_fsm_state = SN)) in
let s_fsm_srs = (((S_fsm_state = SPF) /\ ~S_fsm_rst)
              \/ ((S_fsm_state = ST) /\ ~S_fsm_rst)) in
let s_fsm_scs = (new_S_fsm_state = SCS) in
let new_S_soft_shot = (~ASel(Gcrh) /\ ASel(Gcrl)) in
```

```
let new_S_soft_shot_del = new_S_soft_shot in
let s_soft_cnt_out =
   ((new_S_soft_shot_del /\ ~S_soft_shot_del)
     => (INCN 2 S_soft_cnt)
     | S_soft_cnt) in
let new_S_soft_cnt =
   ((s_fsm_srs) => (WORDN 2 0) |
    (~BSel(Gcrh) /\ ~BSel(Gcrl)) => (RSTN 2) | s_soft_cnt_out) in
let s_delay_out =
   ((s_fsm_sec) => (INCN 17 S_delay) | S_delay) in
let new_S_delay =
   ((s_fsm_src \/ (s_fsm_scs /\ (ELEMENT s_delay_out (6))))
      => (WORDN 17 0)
      | s_delay_out) in
let new_S_instart =
   ((ASel(Test))
    => (ELEMENT s_delay_out (5))
     | (ELEMENT s_delay_out (16))) in
let s_cpu0_ok =
   (s_fsm_sc0f /\ BSel(Failure0_) /\ (s_soft_cnt_out = (WORDN 2 5))) in
let s_cpu1_ok =
   (s_fsm_sc1f /\ BSel(Failure1_) /\ (s_soft_cnt_out = (WORDN 2 5))) in
let new_S_pmm_fail =
   ((s_fsm_sb \/ s_fsm_spmf)
    => ((s_fsm_sb /\ ~s_fsm_spmf) => T |
        (~s_fsm_sb /\ s_fsm_spmf) => F |
        (~s_fsm_sb /\ ~s_fsm_spmf) => F | ARB)
    | S_pmm_fail) in
let new_S_cpu0_fail =
   ((s_fsm_sb \/ BSel(Bypass) \/ s_cpu0_ok)
    => ((s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu0_ok)) => T |
        (~s_fsm_sb /\ (BSel(Bypass) \/ s_cpu0_ok)) => F |
        (~s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu0_ok)) => F | ARB)
    | S_cpu0_fail) in
let new_S_cpu1_fail =
   ((s_fsm_sb \/ BSel(Bypass) \/ s_cpu1_ok)
    => ((s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu1_ok)) => T |
        (~s_fsm_sb /\ (BSel(Bypass) \/ s_cpu1_ok)) => F |
        (~s_fsm_sb /\ ~(BSel(Bypass) \/ s_cpu1_ok)) => F | ARB)
    | S_cpu1_fail) in
let new_S_piu_fail =
   ((s_fsm_sb \/ BSel(Bypass) \/ s_fsm_spf)
    => ((s_fsm_sb /\ ~(BSel(Bypass) \/ s_fsm_spf)) => T |
        (~s_fsm_sb /\ (BSel(Bypass) \/ s_fsm_spf)) => F |
        (~s_fsm_sb /\ ~(BSel(Bypass) \/ s_fsm_spf)) => F | ARB)
    | S_piu_fail) in
let s_cpu0_select = ((s_fsm_sn \/ s_fsm_so) /\ ~new_S_cpu0_fail) in
let s_cpu1_select =
   (~new_S_cpu1_fail /\ (s_fsm_sn \/ s_fsm_so) /\ new_S_cpu0_fail) in
let new_S_bad_cpu0 =
   ((s_cpu0_select \/ s_fsm_sb)
    => ((s_fsm_sb /\ ~s_cpu0_select) => T |
        (~s_fsm_sb /\ s_cpu0_select) => F |
        (~s_fsm_sb /\ ~s_cpu0_select) => F | ARB)
    | S_bad_cpu0) in
let new_S_bad_cpu1 =
   ((s_cpu1_select \/ s_fsm_sb)
    => ((s_fsm_sb /\ ~s_cpu1_select) => T |
        (~s_fsm_sb /\ s_cpu1_select) => F |
        (~s_fsm_sb /\ ~s_cpu1_select) => F | ARB)
    | S_bad_cpu1) in
let new_S_reset_cpu0 = (new_S_bad_cpu0 /\ s_fsm_src0) in
let new_S_reset_cpu1 = (new_S_bad_cpu1 /\ s_fsm_src1) in
let new_S_cpu_bist =
   (S_reset_cpu0 /\ S_reset_cpu1 /\ ASel(Bypass)) in
let new_S_fsm_rst = BSel(Rst) in
let new_S_fsm_delay6 = (ELEMENT s_delay_out (6)) in
let new_S_fsm_delay17 =
   ((BSel(Test))
    => (ELEMENT s_delay_out (6))
     | (ELEMENT s_delay_out (17))) in
let new_S_fsm_bothbad = (new_S_cpu0_fail /\ new_S_cpu1_fail) in
```

```
        let new_S_fsm_bypass = BSel(Bypass) in

        let ss0 =
           (ALTER
              ARBN
              (0)
              ((new_S_fsm_state = SRA) \/ (new_S_fsm_state = SPF) \/
               (new_S_fsm_state = ST) \/ (new_S_fsm_state = SC1I) \/
               (new_S_fsm_state = SCS) \/ (new_S_fsm_state = SN) \/
               (new_S_fsm_state = SO))) in
        let ss1 =
           (ALTER
              ss0
              (1)
              ((new_S_fsm_state = SPF) \/ (new_S_fsm_state = SC0I) \/
               (new_S_fsm_state = SC0F) \/ (new_S_fsm_state = ST) \/
               (new_S_fsm_state = SSTOP) \/ (new_S_fsm_state = SO))) in
        let ss2 =
           (ALTER
              ss1
              (2)
              ((new_S_fsm_state = SC0F) \/ (new_S_fsm_state = ST) \/
               (new_S_fsm_state = SC1I) \/ (new_S_fsm_state = SC1F) \/
               (new_S_fsm_state = SS) \/ (new_S_fsm_state = SSTOP) \/
               (new_S_fsm_state = SCS))) in
        let ss3 =
           (ALTER
              ss2
              (3)
              ((new_S_fsm_state = SS) \/ (new_S_fsm_state = SSTOP) \/
               (new_S_fsm_state = SCS) \/ (new_S_fsm_state = SN) \/
               (new_S_fsm_state = SO))) in
        let S_state = (ss3, ss3) in
        let Reset_cport = (s_fsm_srcp, s_fsm_srcp) in
        let Disable_int =
           ((~new_S_instart /\
             (~s_fsm_sn \/ ~(ELEMENT s_delay_out (6))) /\
             s_fsm_sdi),
            (~new_S_instart /\
             (~s_fsm_sn \/ ~(ELEMENT s_delay_out (6))) /\
             s_fsm_sdi)) in
        let Reset_piu = (s_fsm_srp, s_fsm_srp) in
        let Reset_cpu0 = (S_reset_cpu0, new_S_reset_cpu0) in
        let Reset_cpu1 = (S_reset_cpu1, new_S_reset_cpu1) in
        let Cpu_bist = (S_cpu_bist, new_S_cpu_bist) in
        let Piu_fail = (S_piu_fail, new_S_piu_fail) in
        let Cpu0_fail = (S_cpu0_fail, new_S_cpu0_fail) in
        let Cpu1_fail = (S_cpu1_fail, new_S_cpu1_fail) in
        let Pmm_fail = (S_pmm_fail, new_S_pmm_fail) in

        (SCOut S_state Reset_cport Disable_int Reset_piu Reset_cpu0 Reset_cpu1
             Cpu_bist Piu_fail Cpu0_fail Cpu1_fail Pmm_fail)"
   );;

let SClockOF_REW = save_thm
   ('SClockOF_REW',
    (REWRITE_RULE [ASel;BSel;RSTN] SC_OF)
   );;

let SC_Exec = new_definition
   ('SC_Exec',
    "! (sci :SCI) (s :timeC->s_state) (e :timeC->s_env) (p :timeC->s_out)
       (t :timeC) .
     SC_Exec sci s e p t = T"
   );;

let SC_PreC = new_definition
   ('SC_PreC',
    "! (sci :SCI) (s :timeC->s_state) (e :timeC->s_env) (p :timeC->s_out)
       (t :timeC) .
     SC_PreC sci s e p t = T"
   );;
```

```
let SC_PostC = new_definition
   ('SC_PostC',
    "! (sci :SCI) (s :timeC->s_state) (e :timeC->s_env) (p :timeC->s_out)
       (t :timeC) .
     SC_PostC sci s e p t =
        (s (t+1) = SC_NSF (s t) (e t)) /\
        (p t = SC_OF (s t) (e t))"
   );;

let SC_Correct = new_definition
   ('SC_Correct',
    "! (sci :SCI) (s :timeC->s_state) (e :timeC->s_env) (p :timeC->s_out)
       (t :timeC) .
     SC_Correct sci s e p t =
        SC_Exec sci s e p t /\
        SC_PreC sci s e p t
        ==>
        SC_PostC sci s e p t"
   );;

let SCSet_Correct = new_definition
   ('SCSet_Correct',
    "! (s :timeC->s_state) (e :timeC->s_env) (p :timeC->s_out) .
     SCSet_Correct s e p = !(sci:SCI)(t:timeC). SC_Correct sci s e p t"
   );;

close_theory();;
```

4 PIU Requirements Specification

This section contains the HOL listings for major portions of the PIU requirements specification. Specifically, it contains most of the definition for the PIU behavior associated with memory accesses initiated by the local processor.

Subsection 4.1 contains the transaction-level specification for the PIU's handling of local-processor-initiated memory accesses. It contains two theories—*piutaux_def* defines PIU-level data structures and *piutransp_def* defines the PIU behavior itself.

Subsections 4.2–4.5 contain the transaction-level specifications for the P-Port, M-Port, C-Port, and R-Port, respectively. Each subsection contains two theories, defining the port-level data structures and the specifications themselves. Subsections 4.2 and 4.3 contain the abstraction predicates for the P-Port and M-Port, respectively.

4.1 PIU Transaction-Level Specification

This section contains the theories *piutauxp_def* and *piutransp_def*, defining the PIU transaction-level data structures and interpreter.

```
%-----------------------------------------------------------------------------

   File:        piutauxp_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        2 March 1993

   This file contains types and definitions for the transaction-level
   specification of the PIU P-Process.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                 ]);;

set_flag ('timing', true);;

system 'rm piutauxp_def.th';;

new_theory 'piutauxp_def';;

map new_parent ['wordn_def';'array_def';'ineq';'piuaux_def'];;

new_type_abbrev ('wordn',":num->bool");;
new_type_abbrev ('wordnn',":num->wordn");;

%-----------------------------------------------------------------------------
   Abstract data type for the PIU P-Process instruction opcodes.
-----------------------------------------------------------------------------%

let PI =
   define_type 'PI'
               'PI = PWriteLM | PReadLM | PWritePIU | PReadPIU | PWriteCB |
                     PReadCB';;

%-----------------------------------------------------------------------------
   Abstract data type for the PIU transaction opcodes.
-----------------------------------------------------------------------------%

% P-Bus Master Opcodes %
```

```
let pbmop =
   define_type 'pbmop'
               'pbmop = PBM_WriteLM | PBM_WritePIU | PBM_WriteCB | PBM_ReadLM |
                        PBM_ReadPIU | PBM_ReadCB | PBM_Illegal';;

% P-Bus Slave Opcodes %
let pbsop =
   define_type 'pbsop'
               'pbsop = PBS_Ready | PBS_Illegal';;

% M-Bus Master Opcodes %
let mbmop =
   define_type 'mbmop'
               'mbmop = MBM_WriteLM | MBM_ReadLM | MBM_Idle | MBM_Illegal';;

% M-Bus Slave Opcodes %
let mbsop =
   define_type 'mbsop'
               'mbsop = MBS_Ready | MBS_Illegal';;

% C-Bus Master Opcodes %
let cbmop =
   define_type 'cbmop'
               'cbmop = CBM_WriteCB | CBM_ReadCB | CBM_Idle | CBM_Illegal';;

% C-Bus Slave Opcodes %
let cbsop =
   define_type 'cbsop'
               'cbsop = CBS_Ready | CBS_Illegal';;

% I-Bus Slave Opcodes %
let ibsop =
   define_type 'ibsop'
               'ibsop = IBS_Ready | IBS_Idle | IBS_Illegal';;

% I-Bus Arbitration-Master Opcodes %
let ibamop =
   define_type 'ibamop'
               'ibamop = IBAM_ProcP | IBAM_ProcP | IBAM_Illegal';;

% I-Bus Arbitration-Slave Opcodes %
let ibasop =
   define_type 'ibasop'
               'ibasop = IBAS_Ready | IBAS_Illegal';;

% Environment-Reset Master Opcodes %
let ermop =
   define_type 'ermop'
               'ermop = ERM_NoReset | ERM_Illegal';;

% Internal-Reset Master Opcodes %
let rmop =
   define_type 'rmop'
               'rmop = RM_NoReset | RM_Illegal';;

%-----------------------------------------------------------------------------
   Abstract data type for the memory access target.
-----------------------------------------------------------------------------%

let targ_Axiom =
   define_type 'targ_Axiom'
               'targ = LM | PIU | CB';;

%-----------------------------------------------------------------------------
   Abstract data type for the state.
-----------------------------------------------------------------------------%

let piut_state =
   define_type 'piut_state'
               'piut_state = PIUTState wordn wordn wordn wordn wordn wordn wordn
                                      wordn wordn wordn wordn wordn sfsm_ty';;
```

```
let RT_icrS = new_recursive_definition
    false
    piut_state
    'RT_icrS'
    "RT_icrS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_icr";;

let RT_gcrS = new_recursive_definition
    false
    piut_state
    'RT_gcrS'
    "RT_gcrS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_gcr";;

let RT_ccrS = new_recursive_definition
    false
    piut_state
    'RT_ccrS'
    "RT_ccrS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ccr";;

let RT_srS = new_recursive_definition
    false
    piut_state
    'RT_srS'
    "RT_srS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_sr";;

let RT_ctr0_inS = new_recursive_definition
    false
    piut_state
    'RT_ctr0_inS'
    "RT_ctr0_inS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr0_in";;

let RT_ctr1_inS = new_recursive_definition
    false
    piut_state
    'RT_ctr1_inS'
    "RT_ctr1_inS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr1_in";;

let RT_ctr2_inS = new_recursive_definition
    false
    piut_state
    'RT_ctr2_inS'
    "RT_ctr2_inS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr2_in";;

let RT_ctr3_inS = new_recursive_definition
    false
    piut_state
    'RT_ctr3_inS'
    "RT_ctr3_inS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr3_in";;
```

```
let RT_ctr0S = new_recursive_definition
    false
    piut_state
    'RT_ctr0S'
    "RT_ctr0S (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr0";;

let RT_ctr1S = new_recursive_definition
    false
    piut_state
    'RT_ctr1S'
    "RT_ctr1S (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr1";;

let RT_ctr2S = new_recursive_definition
    false
    piut_state
    'RT_ctr2S'
    "RT_ctr2S (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr2";;

let RT_ctr3S = new_recursive_definition
    false
    piut_state
    'RT_ctr3S'
    "RT_ctr3S (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                        RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3
                        ST_fsm_state)
     = RT_ctr3";;

let ST_fsm_stateS = new_recursive_definition
    false
    piut_state
    'ST_fsm_stateS'
    "ST_fsm_stateS (PIUTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                              RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2
                              RT_ctr3 ST_fsm_state)
     = ST_fsm_state";;

let State_CASES =
   prove_cases_thm (prove_induction_thm piut_state);;

let PIUTState_Selectors_Work = prove_thm
   ('PIUTState_Selectors_Work',
    "! (s :piut_state) .
     s = (PIUTState (RT_icrS s)(RT_gcrS s)(RT_ccrS s)(RT_srS s)(RT_ctr0_inS s)
                    (RT_ctr1_inS s)(RT_ctr2_inS s)(RT_ctr3_inS s)(RT_ctr0S s)
                    (RT_ctr1S s)(RT_ctr2S s)(RT_ctr3S s)(ST_fsm_stateS s))",
   GEN_TAC
   THEN STRUCT_CASES_TAC (SPEC "s:piut_state" State_CASES)
   THEN REWRITE_TAC [RT_icrS;RT_gcrS;RT_ccrS;RT_srS;RT_ctr0_inS;RT_ctr1_inS;
                     RT_ctr2_inS;RT_ctr3_inS;RT_ctr0S;RT_ctr1S;RT_ctr2S;RT_ctr3S;
                     ST_fsm_stateS]
   );;

%-----------------------------------------------------------------------------
   Abstract data type for the environment.
-----------------------------------------------------------------------------%

let piut_env =
   define_type 'piut_env'
               'piut_env = PIUTEnv pbmop wordn wordnn wordn wordnn bool
                                   mbsop wordnn
                                   cbsop wordnn
                                   ermop';;
```

```
let PB_Opcode_inE = new_recursive_definition
    false
    piut_env
    'PB_Opcode_inE'
    "PB_Opcode_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = PB_Opcode_in";;

let PB_Addr_inE = new_recursive_definition
    false
    piut_env
    'PB_Addr_inE'
    "PB_Addr_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = PB_Addr_in";;

let PB_Data_inE = new_recursive_definition
    false
    piut_env
    'PB_Data_inE'
    "PB_Data_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = PB_Data_in";;

let PB_BS_inE = new_recursive_definition
    false
    piut_env
    'PB_BS_inE'
    "PB_BS_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = PB_BS_in";;

let PB_BE_inE = new_recursive_definition
    false
    piut_env
    'PB_BE_inE'
    "PB_BE_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = PB_BE_in";;

let PB_Lock_inE = new_recursive_definition
    false
    piut_env
    'PB_Lock_inE'
    "PB_Lock_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = PB_Lock_in";;

let MB_Opcode_inE = new_recursive_definition
    false
    piut_env
    'MB_Opcode_inE'
    "MB_Opcode_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = MB_Opcode_in";;

let MB_Data_inE = new_recursive_definition
    false
    piut_env
    'MB_Data_inE'
    "MB_Data_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = MB_Data_in";;
```

```
let CB_Opcode_inE = new_recursive_definition
    false
    piut_env
    'CB_Opcode_inE'
    "CB_Opcode_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = CB_Opcode_in";;

let CB_Data_inE = new_recursive_definition
    false
    piut_env
    'CB_Data_inE'
    "CB_Data_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = CB_Data_in";;

let ERM_Reset_inE = new_recursive_definition
    false
    piut_env
    'ERM_Reset_inE'
    "ERM_Reset_inE (PIUTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                            PB_BE_in PB_Lock_in MB_Opcode_in MB_Data_in
                            CB_Opcode_in CB_Data_in ERM_Reset_in)
    = ERM_Reset_in";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm piut_env);;

let PTEnv_Selectors_Work = prove_thm
   ('PTEnv_Selectors_Work',
    "! (e :piut_env) .
     e = (PIUTEnv (PB_Opcode_inE e) (PB_Addr_inE e) (PB_Data_inE e)
                  (PB_BS_inE e) (PB_BE_inE e) (PB_Lock_inE e) (MB_Opcode_inE e)
                  (MB_Data_inE e) (CB_Opcode_inE e) (CB_Data_inE e)
                  (ERM_Reset_inE e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:piut_env" Env_CASES)
    THEN REWRITE_TAC [PB_Opcode_inE; PB_Addr_inE; PB_Data_inE;
                      PB_BS_inE; PB_BE_inE; PB_Lock_inE; MB_Opcode_inE;
                      MB_Data_inE; CB_Opcode_inE; CB_Data_inE; ERM_Reset_inE]
   );;

%-----------------------------------------------------------------------------
   Abstract data type for the output.
-----------------------------------------------------------------------------%

let piut_out =
   define_type 'piut_out'
               'piut_out = PIUTOut pbsop wordnn
                                   mbmop wordnn wordnn wordn
                                   cbmop wordn wordnn wordn wordnn';;

let PB_Opcode_outO = new_recursive_definition
    false
    piut_out
    'PB_Opcode_outO'
    "PB_Opcode_outO (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = PB_Opcode_out";;

let PB_Data_outO = new_recursive_definition
    false
    piut_out
    'PB_Data_outO'
    "PB_Data_outO (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                           MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                           CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = PB_Data_out";;
```

```
let MB_Opcode_out0 = new_recursive_definition
    false
    piut_out
    'MB_Opcode_out0'
    "MB_Opcode_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = MB_Opcode_out";;

let MB_Addr_out0 = new_recursive_definition
    false
    piut_out
    'MB_Addr_out0'
    "MB_Addr_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = MB_Addr_out";;

let MB_Data_out0 = new_recursive_definition
    false
    piut_out
    'MB_Data_out0'
    "MB_Data_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = MB_Data_out";;

let MB_BS_out0 = new_recursive_definition
    false
    piut_out
    'MB_BS_out0'
    "MB_BS_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = MB_BS_out";;

let CB_Opcode_out0 = new_recursive_definition
    false
    piut_out
    'CB_Opcode_out0'
    "CB_Opcode_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = CB_Opcode_out";;

let CB_Addr_out0 = new_recursive_definition
    false
    piut_out
    'CB_Addr_out0'
    "CB_Addr_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = CB_Addr_out";;

let CB_Data_out0 = new_recursive_definition
    false
    piut_out
    'CB_Data_out0'
    "CB_Data_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = CB_Data_out";;

let CB_BS_out0 = new_recursive_definition
    false
    piut_out
    'CB_BS_out0'
    "CB_BS_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                             MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                             CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = CB_BS_out";;
```

```
let CB_BE_out0 = new_recursive_definition
    false
    piut_out
    'CB_BE_out0'
    "CB_BE_out0 (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out
                         MB_Addr_out MB_Data_out MB_BS_out CB_Opcode_out
                         CB_Addr_out CB_Data_out CB_BS_out CB_BE_out)
    = CB_BE_out";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm piut_out);;

let PTOut_Selectors_Work = prove_thm
   ('PTOut_Selectors_Work',
    "! (p :piut_out) .
     p = (PIUTOut (PB_Opcode_out0 p) (PB_Data_out0 p) (MB_Opcode_out0 p)
                  (MB_Addr_out0 p) (MB_Data_out0 p) (MB_BS_out0 p)
                  (CB_Opcode_out0 p) (CB_Addr_out0 p) (CB_Data_out0 p)
                  (CB_BS_out0 p) (CB_BE_out0 p))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "p:piut_out" Out_CASES)
    THEN REWRITE_TAC [PB_Opcode_out0; PB_Data_out0; MB_Opcode_out0;
                      MB_Addr_out0; MB_Data_out0; MB_BS_out0; CB_Opcode_out0;
                      CB_Addr_out0; CB_Data_out0; CB_BS_out0; CB_BE_out0]
   );;

%----------------------------------------------------------------------------
   Memory Target Predicates.
----------------------------------------------------------------------------%

let CBusAddrP = new_definition
   ('CBusAddrP',
    "! (a :wordn) . CBusAddrP a = ELEMENT a (29)"
   );;

let PRegAddrP = new_definition
   ('PRegAddrP',
    "! (a :wordn) .
     PRegAddrP a = ~(ELEMENT a (29)) /\ (SUBARRAY a (23,22) = WORDN 1 3)"
   );;

let LMemAddrP = new_definition
   ('LMemAddrP',
    "! (a :wordn) .
     LMemAddrP a = ~(ELEMENT a (29)) /\ ~(SUBARRAY a (23,22) = WORDN 1 3)"
   );;

let Reg0P = new_definition
   ('Reg0P',
    "! (a :wordn) . Reg0P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 0)"
   );;

let Reg1P = new_definition
   ('Reg1P',
    "! (a :wordn) . Reg1P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 1)"
   );;

let Reg2P = new_definition
   ('Reg2P',
    "! (a :wordn) . Reg2P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 2)"
   );;

let Reg3P = new_definition
   ('Reg3P',
    "! (a :wordn) . Reg3P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 3)"
   );;

let Reg4P = new_definition
   ('Reg4P',
    "! (a :wordn) . Reg4P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 4)"
   );;
```

```
let Reg5P = new_definition
    ('Reg5P',
     "! (a :wordn) . Reg5P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 5)"
    );;

let Reg6P = new_definition
    ('Reg6P',
     "! (a :wordn) . Reg6P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 6)"
    );;

let Reg7P = new_definition
    ('Reg7P',
     "! (a :wordn) . Reg7P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 7)"
    );;

let Reg8P = new_definition
    ('Reg8P',
     "! (a :wordn) . Reg8P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 8)"
    );;

let Reg9P = new_definition
    ('Reg9P',
     "! (a :wordn) . Reg9P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 9)"
    );;

let Reg10P = new_definition
    ('Reg10P',
     "! (a :wordn) . Reg10P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 10)"
    );;

let Reg11P = new_definition
    ('Reg11P',
     "! (a :wordn) . Reg11P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 11)"
    );;

let Reg12P = new_definition
    ('Reg12P',
     "! (a :wordn) . Reg12P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 12)"
    );;

let Reg13P = new_definition
    ('Reg13P',
     "! (a :wordn) . Reg13P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 13)"
    );;

let Reg14P = new_definition
    ('Reg14P',
     "! (a :wordn) . Reg14P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 14)"
    );;

let Reg15P = new_definition
    ('Reg15P',
     "! (a :wordn) . Reg15P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 15)"
    );;

close_theory();;


%----------------------------------------------------------------------------

   File:        piutransp_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        2 March 1993

   This file contains the transaction-level behavioral specification for the
   P-Process for the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.  The P-Process defines memory
   access transactions initiated by the local PMM processor.

----------------------------------------------------------------------------%
```

```
set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/piu/pproc/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing', true);;

system 'rm piutransp_def.th';;

new_theory 'piutransp_def';;

loadf 'abs_theory';;

map new_parent ['piuaux_def';'piutauxp_def';'wordn_def';'array_def';'ineq'];;

new_type_abbrev ('timeT',":num");;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%----------------------------------------------------------------------------
   Next-state function definitions for PIU P-Process.
----------------------------------------------------------------------------%

let PStable_State_NSF = new_definition
   ('PStable_State_NSF',
    "! (s :piut_state) (e :piut_env) .
     PStable_State_NSF s e =
        let new_RT_icr = RT_icrS s in
        let new_RT_gcr = RT_gcrS s in
        let new_RT_ccr = RT_ccrS s in
        let new_RT_sr = (ARBN:wordn) in
        let new_RT_ctr0_in = RT_ctr0_inS s in
        let new_RT_ctr1_in = RT_ctr1_inS s in
        let new_RT_ctr2_in = RT_ctr2_inS s in
        let new_RT_ctr3_in = RT_ctr3_inS s in
        let new_RT_ctr0 = RT_ctr0S s in
        let new_RT_ctr1 = RT_ctr1S s in
        let new_RT_ctr2 = RT_ctr2S s in
        let new_RT_ctr3 = RT_ctr3S s in
        let new_ST_fsm_state = ST_fsm_stateS s in

        (PIUTState new_RT_icr new_RT_gcr new_RT_ccr new_RT_sr new_RT_ctr0_in
                   new_RT_ctr1_in new_RT_ctr2_in new_RT_ctr3_in new_RT_ctr0
                   new_RT_ctr1 new_RT_ctr2 new_RT_ctr3 new_ST_fsm_state)"
   );;

let PWrite_PIU_NSF = new_definition
   ('PWrite_PIU_NSF',
    "! (s :piut_state) (e :piut_env) .
     PWrite_PIU_NSF s e =
        let RT_icr = RT_icrS s and
            RT_gcr = RT_gcrS s and
            RT_ccr = RT_ccrS s and
            RT_ctr0_in = RT_ctr0_inS s and
            RT_ctr1_in = RT_ctr1_inS s and
            RT_ctr2_in = RT_ctr2_inS s and
            RT_ctr3_in = RT_ctr3_inS s and
            RT_ctr0 = RT_ctr0S s and
            RT_ctr1 = RT_ctr1S s and
            RT_ctr2 = RT_ctr2S s and
            RT_ctr3 = RT_ctr3S s in
        let PB_Addr_in = PB_Addr_inE e and
            PB_Data_in = PB_Data_inE e and
            PB_BS_in = PB_BS_inE e and
            PB_BE_in = PB_BE_inE e in
        let new_RT_icr =
           (((Reg0P PB_Addr_in) /\ (VAL 1 PB_BS_in = 0)) =>
                 (ANDN 31 (ELEMENT PB_Data_in (0)) RT_icr) |
            ((Reg15P PB_Addr_in) /\ (VAL 1 PB_BS_in = 1)) =>
                 (ANDN 31 (ELEMENT PB_Data_in (1)) RT_icr) |
```

```
     ((Reg14P PB_Addr_in) /\ (VAL 1 PB_BS_in = 2)) =>
          (ANDN 31 (ELEMENT PB_Data_in (2)) RT_icr) |
     ((Reg13P PB_Addr_in) /\ (VAL 1 PB_BS_in = 3)) =>
          (ANDN 31 (ELEMENT PB_Data_in (3)) RT_icr) |
     (Reg1P PB_Addr_in) =>
        (ORN 31 (ELEMENT PB_Data_in (0)) RT_icr) |
     ((Reg0P PB_Addr_in) /\ (VAL 1 PB_BS_in >= 1)) =>
          (ANDN 31 (ELEMENT PB_Data_in (1))
                   (ORN 31 (ELEMENT PB_Data_in (0)) RT_icr)) |
     ((Reg15P PB_Addr_in) /\ (VAL 1 PB_BS_in >= 2)) =>
          (ANDN 31 (ELEMENT PB_Data_in (2))
                   (ORN 31 (ELEMENT PB_Data_in (1)) RT_icr)) |
     ((Reg14P PB_Addr_in) /\ (VAL 1 PB_BS_in >= 3)) =>
          (ANDN 31 (ELEMENT PB_Data_in (3))
                   (ORN 31 (ELEMENT PB_Data_in (2)) RT_icr)) | RT_icr) in
let new_RT_gcr =
    ((Reg2P PB_Addr_in) => ELEMENT PB_Data_in (0) |
     (Reg1P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
     (Reg0P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
     (Reg15P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_gcr) in
let new_RT_ccr =
    ((Reg3P PB_Addr_in) => ELEMENT PB_Data_in (0) |
     (Reg2P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
     (Reg1P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
     (Reg0P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ccr) in
let new_RT_sr = (ARBN:wordn) in
let new_RT_ctr0_in =
    ((Reg8P PB_Addr_in) => ELEMENT PB_Data_in (0) |
     (Reg7P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
     (Reg6P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
     (Reg5P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr0_in) in
let new_RT_ctr1_in =
    ((Reg9P PB_Addr_in) => ELEMENT PB_Data_in (0) |
     (Reg8P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
     (Reg7P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
     (Reg6P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr1_in) in
let new_RT_ctr2_in =
    ((Reg10P PB_Addr_in) => ELEMENT PB_Data_in (0) |
     (Reg9P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
     (Reg8P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
     (Reg7P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr2_in) in
let new_RT_ctr3_in =
    ((Reg11P PB_Addr_in) => ELEMENT PB_Data_in (0) |
     (Reg10P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
     (Reg9P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
     (Reg8P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr3_in) in
let new_RT_ctr0 =
    ((Reg12P PB_Addr_in) => ELEMENT PB_Data_in (0) |
     (Reg11P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
     (Reg10P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
     (Reg9P PB_Addr_in /\
        (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr0) in
```

```
        let new_RT_ctr1 =
           ((Reg13P PB_Addr_in) => ELEMENT PB_Data_in (0) |
            (Reg12P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
            (Reg11P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
            (Reg10P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr1) in
        let new_RT_ctr2 =
           ((Reg14P PB_Addr_in) => ELEMENT PB_Data_in (0) |
            (Reg13P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
            (Reg12P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 2)) =>ELEMENT PB_Data_in (2) |
            (Reg11P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr2) in
        let new_RT_ctr3 =
           ((Reg15P PB_Addr_in) => ELEMENT PB_Data_in (0) |
            (Reg14P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 1)) => ELEMENT PB_Data_in (1) |
            (Reg13P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 2)) => ELEMENT PB_Data_in (2) |
            (Reg12P PB_Addr_in /\
               (VAL 1 PB_BS_in >= 3)) => ELEMENT PB_Data_in (3) | RT_ctr3) in
        let new_ST_fsm_state = ST_fsm_stateS s in

        (PIUTState new_RT_icr new_RT_gcr new_RT_ccr new_RT_sr new_RT_ctr0_in
                   new_RT_ctr1_in new_RT_ctr2_in new_RT_ctr3_in new_RT_ctr0
                   new_RT_ctr1 new_RT_ctr2 new_RT_ctr3 new_ST_fsm_state)"
    );;

%-----------------------------------------------------------------------------
   Output function definitions for the PIU P-Process instructions.
-----------------------------------------------------------------------------%

let PWriteLM_OF = new_definition
    ('PWriteLM_OF',
     "! (rep :^REP_ty) (s :piut_state) (e :piut_env) .
      PWriteLM_OF rep s e =
         let PB_Opcode_out = PBS_Ready in
         let PB_Data_out = (ARBN:num->wordn) in
         let MB_Opcode_out = MBM_WriteLM in

         let bs = VAL 1 (PB_BS_inE e) in
         let a0 = PB_Addr_inE e in
         let a0_0 = ALTER ARBN (0) a0 in
         let a1_0 = ALTER a0_0 (1) (bs > 0 => (INCN 18 a0) | ARBN) in
         let a2_0 = ALTER a1_0 (2) (bs > 1 => (INCN 18 (INCN 18 a0)) | ARBN) in
         let a3_0 = ALTER a2_0 (3) (bs > 2 => (INCN 18 (INCN 18 (INCN 18 a0)))
                                              | ARBN) in
         let MB_Addr_out = a3_0 in

         let d0 = ELEMENT (PB_Data_inE e) (0) in
         let d1 = ELEMENT (PB_Data_inE e) (1) in
         let d2 = ELEMENT (PB_Data_inE e) (2) in
         let d3 = ELEMENT (PB_Data_inE e) (3) in
         let m0 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (0)) in
         let m1 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (1)) in
         let m2 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (2)) in
         let m3 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (3)) in
         let be0 = ELEMENT (PB_BE_inE e) (0) in
         let be1 = ELEMENT (PB_BE_inE e) (1) in
         let be2 = ELEMENT (PB_BE_inE e) (2) in
         let be3 = ELEMENT (PB_BE_inE e) (3) in
         let o00 = ELEMENT be0 (0) => SUBARRAY d0 (7,0) | SUBARRAY m0 (7,0) in
         let o01 = ELEMENT be0 (1) => SUBARRAY d0 (15,8) | SUBARRAY m0 (15,8) in
         let o02 = ELEMENT be0 (2) => SUBARRAY d0 (23,16) | SUBARRAY m0 (23,16) in
         let o03 = ELEMENT be0 (3) => SUBARRAY d0 (31,24) | SUBARRAY m0 (31,24) in
         let o10 = ELEMENT be1 (0) => SUBARRAY d1 (7,0) | SUBARRAY m1 (7,0) in
         let o11 = ELEMENT be1 (1) => SUBARRAY d1 (15,8) | SUBARRAY m1 (15,8) in
         let o12 = ELEMENT be1 (2) => SUBARRAY d1 (23,16) | SUBARRAY m1 (23,16) in
         let o13 = ELEMENT be1 (3) => SUBARRAY d1 (31,24) | SUBARRAY m1 (31,24) in
```

```
        let o20 = ELEMENT be2 (0) => SUBARRAY d2 (7,0) | SUBARRAY m2 (7,0) in
        let o21 = ELEMENT be2 (1) => SUBARRAY d2 (15,8) | SUBARRAY m2 (15,8) in
        let o22 = ELEMENT be2 (2) => SUBARRAY d2 (23,16) | SUBARRAY m2 (23,16) in
        let o23 = ELEMENT be2 (3) => SUBARRAY d2 (31,24) | SUBARRAY m2 (31,24) in
        let o30 = ELEMENT be3 (0) => SUBARRAY d3 (7,0) | SUBARRAY m3 (7,0) in
        let o31 = ELEMENT be3 (1) => SUBARRAY d3 (15,8) | SUBARRAY m3 (15,8) in
        let o32 = ELEMENT be3 (2) => SUBARRAY d3 (23,16) | SUBARRAY m3 (23,16) in
        let o33 = ELEMENT be3 (3) => SUBARRAY d3 (31,24) | SUBARRAY m3 (31,24) in
        let d00 = MALTER ARBN (7,0) o00 in
        let d01 = MALTER d00 (15,8) o01 in
        let d02 = MALTER d01 (23,16) o02 in
        let d03 = MALTER d02 (31,24) o03 in
        let d10 = MALTER ARBN (7,0) o10 in
        let d11 = MALTER d10 (15,8) o11 in
        let d12 = MALTER d11 (23,16) o12 in
        let d13 = MALTER d12 (31,24) o13 in
        let d20 = MALTER ARBN (7,0) o20 in
        let d21 = MALTER d20 (15,8) o21 in
        let d22 = MALTER d21 (23,16) o22 in
        let d23 = MALTER d22 (31,24) o23 in
        let d30 = MALTER ARBN (7,0) o30 in
        let d31 = MALTER d30 (15,8) o31 in
        let d32 = MALTER d31 (23,16) o32 in
        let d33 = MALTER d32 (31,24) o33 in
        let o0_0 = ALTER ARBN (0) (Ham_Enc rep d03) in
        let o1_0 = ALTER o0_0 (1) (bs > 0 => (Ham_Enc rep d13) | ARBN) in
        let o2_0 = ALTER o1_0 (2) (bs > 1 => (Ham_Enc rep d23) | ARBN) in
        let o3_0 = ALTER o2_0 (3) (bs > 2 => (Ham_Enc rep d33) | ARBN) in
        let MB_Data_out = o3_0 in

        let MB_BS_out = PB_BS_inE e in

        let CB_Opcode_out = CBM_Idle in
        let CB_Addr_out = (ARBN:wordn) in
        let CB_Data_out = (ARBN:num->wordn) in
        let CB_BS_out = (ARBN:wordn) in
        let CB_BE_out = (ARBN:num->wordn) in

        (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out MB_Addr_out MB_Data_out
                 MB_BS_out CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                 CB_BE_out)"
    );;

let PReadLM_OF = new_definition
    ('PReadLM_OF',
     "! (rep :^REP_ty) (s :piut_state) (e :piut_env) .
      PReadLM_OF rep s e =
        let PB_Opcode_out = PBS_Ready in
        let PB_Data_out = MB_Data_inE e in
        let MB_Opcode_out = MBM_ReadLM in
        let bs = VAL 1 (PB_BS_inE e) in
        let a0 = PB_Addr_inE e in
        let a0_0 = ALTER ARBN (0) a0 in
        let a1_0 = ALTER a0_0 (1) (bs > 0 => (INCN 18 a0) | ARBN) in
        let a2_0 = ALTER a1_0 (2) (bs > 1 => (INCN 18 (INCN 18 a0)) | ARBN) in
        let a3_0 = ALTER a2_0 (3) (bs > 2 => (INCN 18 (INCN 18 (INCN 18 a0)))
                                             | ARBN) in
        let MB_Addr_out = a3_0 in
        let MB_Data_out = (ARBN:num->wordn) in
        let MB_BS_out = PB_BS_inE e in
        let CB_Opcode_out = CBM_Idle in
        let CB_Addr_out = (ARBN:wordn) in
        let CB_Data_out = (ARBN:num->wordn) in
        let CB_BS_out = (ARBN:wordn) in
        let CB_BE_out = (ARBN:num->wordn) in

        (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out MB_Addr_out MB_Data_out
                 MB_BS_out CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                 CB_BE_out)"
    );;

let PWritePIU_OF = new_definition
```

```
   ('PWritePIU_OF',
    "! (rep :^REP_ty) (s :piut_state) (e :piut_env) .
     PWritePIU_OF rep s e =
        let PB_Opcode_out = PBS_Ready in
        let PB_Data_out = (ARBN:num->wordn) in
        let MB_Opcode_out = MBM_Idle in
        let MB_Addr_out = (ARBN:num->wordn) in
        let MB_Data_out = (ARBN:num->wordn) in
        let MB_BS_out = (ARBN:wordn) in
        let CB_Opcode_out = CBM_Idle in
        let CB_Addr_out = (ARBN:wordn) in
        let CB_Data_out = (ARBN:num->wordn) in
        let CB_BS_out = (ARBN:wordn) in
        let CB_BE_out = (ARBN:num->wordn) in

        (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out MB_Addr_out MB_Data_out
                 MB_BS_out CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                 CB_BE_out)"
   );;

let PReadPIU_OF = new_definition
   ('PReadPIU_OF',
    "! (rep :^REP_ty) (s :piut_state) (e :piut_env) .
     PReadPIU_OF rep s e =
        let PB_Opcode_out = PBS_Ready in
        let bs = VAL 1 (PB_BS_inE e) in
        let d0 = (Reg0P (PB_Addr_inE e)) => RT_icrS s |
                 (Reg1P (PB_Addr_inE e)) => RT_icrS s |
                 (Reg2P (PB_Addr_inE e)) => RT_gcrS s |
                 (Reg3P (PB_Addr_inE e)) => RT_ccrS s |
                 (Reg4P (PB_Addr_inE e)) => RT_srS s |
                 (Reg8P (PB_Addr_inE e)) => RT_ctr0_inS s |
                 (Reg9P (PB_Addr_inE e)) => RT_ctr1_inS s |
                 (Reg10P (PB_Addr_inE e)) => RT_ctr2_inS s |
                 (Reg11P (PB_Addr_inE e)) => RT_ctr3_inS s |
                 (Reg12P (PB_Addr_inE e)) => RT_ctr0S s |
                 (Reg13P (PB_Addr_inE e)) => RT_ctr1S s |
                 (Reg14P (PB_Addr_inE e)) => RT_ctr2S s |
                 (Reg15P (PB_Addr_inE e)) => RT_ctr3S s | ARBN in
        let d1 = (bs > 0)
                  => (Reg15P (PB_Addr_inE e)) => RT_icrS s |
                     (Reg0P (PB_Addr_inE e)) => RT_icrS s |
                     (Reg1P (PB_Addr_inE e)) => RT_gcrS s |
                     (Reg2P (PB_Addr_inE e)) => RT_ccrS s |
                     (Reg3P (PB_Addr_inE e)) => RT_srS s |
                     (Reg7P (PB_Addr_inE e)) => RT_ctr0_inS s |
                     (Reg8P (PB_Addr_inE e)) => RT_ctr1_inS s |
                     (Reg9P (PB_Addr_inE e)) => RT_ctr2_inS s |
                     (Reg10P (PB_Addr_inE e)) => RT_ctr3_inS s |
                     (Reg11P (PB_Addr_inE e)) => RT_ctr0S s |
                     (Reg12P (PB_Addr_inE e)) => RT_ctr1S s |
                     (Reg13P (PB_Addr_inE e)) => RT_ctr2S s |
                     (Reg14P (PB_Addr_inE e)) => RT_ctr3S s | ARBN
                   | ARBN  in
        let d2 = (bs > 1)
                  => (Reg14P (PB_Addr_inE e)) => RT_icrS s |
                     (Reg15P (PB_Addr_inE e)) => RT_icrS s |
                     (Reg0P (PB_Addr_inE e)) => RT_gcrS s |
                     (Reg1P (PB_Addr_inE e)) => RT_ccrS s |
                     (Reg2P (PB_Addr_inE e)) => RT_srS s |
                     (Reg6P (PB_Addr_inE e)) => RT_ctr0_inS s |
                     (Reg7P (PB_Addr_inE e)) => RT_ctr1_inS s |
                     (Reg8P (PB_Addr_inE e)) => RT_ctr2_inS s |
                     (Reg9P (PB_Addr_inE e)) => RT_ctr3_inS s |
                     (Reg10P (PB_Addr_inE e)) => RT_ctr0S s |
                     (Reg11P (PB_Addr_inE e)) => RT_ctr1S s |
                     (Reg12P (PB_Addr_inE e)) => RT_ctr2S s |
                     (Reg13P (PB_Addr_inE e)) => RT_ctr3S s | ARBN
                   | ARBN  in
        let d3 = (bs > 2)
                  => (Reg13P (PB_Addr_inE e)) => RT_icrS s |
                     (Reg14P (PB_Addr_inE e)) => RT_icrS s |
```

```
                    (Reg15P (PB_Addr_inE e)) => RT_gcrS s |
                    (Reg0P (PB_Addr_inE e)) => RT_ccrS s |
                    (Reg1P (PB_Addr_inE e)) => RT_srS s |
                    (Reg5P (PB_Addr_inE e)) => RT_ctr0_inS s |
                    (Reg6P (PB_Addr_inE e)) => RT_ctr1_inS s |
                    (Reg7P (PB_Addr_inE e)) => RT_ctr2_inS s |
                    (Reg8P (PB_Addr_inE e)) => RT_ctr3_inS s |
                    (Reg9P (PB_Addr_inE e)) => RT_ctr0S s |
                    (Reg10P (PB_Addr_inE e)) => RT_ctr1S s |
                    (Reg11P (PB_Addr_inE e)) => RT_ctr2S s |
                    (Reg12P (PB_Addr_inE e)) => RT_ctr3S s | ARBN
                  | ARBN  in
        let PB_Data_out =
              ALTER (ALTER (ALTER (ALTER ARBN(0) d0)(1) d1)(2) d2)(3) d3 in
        let MB_Opcode_out = MBM_Idle in
        let MB_Addr_out = (ARBN:num->wordn) in
        let MB_Data_out = (ARBN:num->wordn) in
        let MB_BS_out = (ARBN:wordn) in
        let CB_Opcode_out = CBM_Idle in
        let CB_Addr_out = (ARBN:wordn) in
        let CB_Data_out = (ARBN:num->wordn) in
        let CB_BS_out = (ARBN:wordn) in
        let CB_BE_out = (ARBN:num->wordn) in

        (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out MB_Addr_out MB_Data_out
                 MB_BS_out CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                 CB_BE_out)"
   );;

let PWriteCB_OF = new_definition
   ('PWriteCB_OF',
    "! (rep :^REP_ty) (s :piut_state) (e :piut_env) .
     PWriteCB_OF rep s e =
        let PB_Opcode_out = PBS_Ready in
        let PB_Data_out = (ARBN:num->wordn) in
        let MB_Opcode_out = MBM_Idle in
        let MB_Addr_out = (ARBN:num->wordn) in
        let MB_Data_out = (ARBN:num->wordn) in
        let MB_BS_out = (ARBN:wordn) in
        let CB_Opcode_out = CBM_WriteCB in
        let CB_Addr_out = PB_Addr_inE e in
        let bs = VAL 1 (PB_BS_inE e) in
        let d0 = ELEMENT (PB_Data_inE e) (0) in
        let d1 = ELEMENT (PB_Data_inE e) (1) in
        let d2 = ELEMENT (PB_Data_inE e) (2) in
        let d3 = ELEMENT (PB_Data_inE e) (3) in
        let o0 = ALTER ARBN (0) (Par_Enc rep d0) in
        let o1 = ALTER o0 (1) (bs > 0 => (Par_Enc rep d1) | ARBN) in
        let o2 = ALTER o1 (2) (bs > 1 => (Par_Enc rep d2) | ARBN) in
        let o3 = ALTER o2 (3) (bs > 2 => (Par_Enc rep d3) | ARBN) in
        let CB_Data_out = o3 in
        let CB_BS_out = PB_BS_inE e in
        let CB_BE_out = PB_BE_inE e in

        (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out MB_Addr_out MB_Data_out
                 MB_BS_out CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                 CB_BE_out)"
   );;

let PReadCB_OF = new_definition
   ('PReadCB_OF',
    "! (rep :^REP_ty) (s :piut_state) (e :piut_env) .
     PReadCB_OF rep s e =
        let PB_Opcode_out = PBS_Ready in
        let bs = VAL 1 (PB_BS_inE e) in
        let d0 = ELEMENT (CB_Data_inE e) (0) in
        let d1 = ELEMENT (CB_Data_inE e) (1) in
        let d2 = ELEMENT (CB_Data_inE e) (2) in
        let d3 = ELEMENT (CB_Data_inE e) (3) in
        let o0 = ALTER ARBN (0) (Par_Dec rep d0) in
        let o1 = ALTER o0 (1) (bs > 0 => (Par_Dec rep d1) | ARBN) in
        let o2 = ALTER o1 (2) (bs > 1 => (Par_Dec rep d2) | ARBN) in
```

```
        let o3 = ALTER o2 (3) (bs > 2 => (Par_Dec rep d3) | ARBN) in
        let PB_Data_out = o3 in
        let MB_Opcode_out = MBM_Idle in
        let MB_Addr_out = (ARBN:num->wordn) in
        let MB_Data_out = (ARBN:num->wordn) in
        let MB_BS_out = (ARBN:wordn) in
        let CB_Opcode_out = CBM_ReadCB in
        let CB_Addr_out = PB_Addr_inE e in
        let CB_Data_out = (ARBN:num->wordn) in
        let CB_BS_out = PB_BS_inE e in
        let CB_BE_out = PB_BE_inE e in

        (PIUTOut PB_Opcode_out PB_Data_out MB_Opcode_out MB_Addr_out MB_Data_out
                 MB_BS_out CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                 CB_BE_out)"
   );;

%-----------------------------------------------------------------------------
   PIU P-Process interpreter definition.
-----------------------------------------------------------------------------%

let PIUP_Exec = new_definition
   ('PIUP_Exec',
    "! (pi :PI) (s :timeT->piut_state) (e :timeT->piut_env) (p :timeT->piut_out)
       (t :timeT) .
     PIUP_Exec pi s e p t =
       (ERM_Reset_inE (e t) = ERM_NoReset) /\
       ((pi = PWriteLM) => (PB_Opcode_inE (e t) = PBM_WriteLM) |
        (pi = PReadLM) => (PB_Opcode_inE (e t) = PBM_ReadLM) |
        (pi = PWritePIU) => (PB_Opcode_inE (e t) = PBM_WritePIU) |
        (pi = PReadPIU) => (PB_Opcode_inE (e t) = PBM_ReadPIU) |
        (pi = PWriteCB) => (PB_Opcode_inE (e t) = PBM_WriteCB)
      % (pi = PReadCB) % | (PB_Opcode_inE (e t) = PBM_ReadCB))"
   );;

let PIUP_PreC = new_definition
   ('PIUP_PreC',
    "! (pi :PI) (s :timeT->piut_state) (e :timeT->piut_env) (p :timeT->piut_out)
       (t :timeT) .
     PIUP_PreC pi s e p t = (ST_fsm_stateS (s t) = SO)"
   );;

let PIUP_PostC = new_definition
   ('PIUP_PostC',
    "! (rep :^REP_ty) (pi :PI) (s :timeT->piut_state) (e :timeT->piut_env)
       (p :timeT->piut_out) (t :timeT) .
     PIUP_PostC rep pi s e p t =
        (pi = PWriteLM) =>  ((s (t+1) = PStable_State_NSF (s t) (e t)) /\
                             (p t = PWriteLM_OF rep (s t) (e t))) |
        (pi = PReadLM) =>   ((s (t+1) = PStable_State_NSF (s t) (e t)) /\
                             (p t = PReadLM_OF rep (s t) (e t))) |
        (pi = PWritePIU) =>  ((s (t+1) = PWrite_PIU_NSF (s t) (e t)) /\
                              (p t = PWritePIU_OF rep (s t) (e t))) |
        (pi = PReadPIU) =>  ((s (t+1) = PStable_State_NSF (s t) (e t)) /\
                             (p t = PReadPIU_OF rep (s t) (e t))) |
        (pi = PWriteCB) =>  ((s (t+1) = PStable_State_NSF (s t) (e t)) /\
                             (p t = PWriteCB_OF rep (s t) (e t))) |
      % (pi = PReadCB) => % ((s (t+1) = PStable_State_NSF (s t) (e t)) /\
                             (p t = PReadCB_OF rep (s t) (e t)))"
   );;

let PIUP_Correct = new_definition
   ('PIUP_Correct',
    "! (rep :^REP_ty) (pi :PI) (s :timeT->piut_state) (e :timeT->piut_env)
       (p :timeT->piut_out) (t :timeT) .
     PIUP_Correct rep pi s e p t =
        PIUP_Exec pi s e p t /\
        PIUP_PreC pi s e p t
        ==>
        PIUP_PostC rep pi s e p t"
   );;
```

```
let PIUPSet_Correct = new_definition
   ('PIUPSet_Correct',
    "! (rep :^REP_ty) (s :timeT->piut_state) (e :timeT->piut_env) .
     PIUPSet_Correct rep s e p = !(pi:PI)(t:timeT). PIUP_Correct rep pi s e p t"
   );;

close_theory();;
```

4.2 P-Port Transaction-Level Specification

This section contains the theories *ptauxp_def*, *ptransp_def*, and *ptabs_def*, defining the P-Port transaction-level data structures, interpreter, and abstraction, respectively.

```
%-----------------------------------------------------------------------------

   File:      ptauxp_def.ml

   Author:    (c) D.A. Fura 1992-93

   Date:      3 March 1993

   This file contains types and definitions for the transaction-level
   specification of the P-Process of the PIU P-Port.

-----------------------------------------------------------------------------%

set_flag ('timing',true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/ftep/piu/hol/pport/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm ptauxp_def.th';;

new_theory 'ptauxp_def';;

map new_parent ['paux_def';'ineq'];;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('timeT', ":num");;
new_type_abbrev ('wordn', ":num->bool");;
new_type_abbrev ('wordnn', ":num->wordn");;

%-----------------------------------------------------------------------------
   Abstract data type for the P-port instruction set.
-----------------------------------------------------------------------------%

let PTI =
   define_type 'PTI'
               'PTI = PT_Write | PT_Read';;

%-----------------------------------------------------------------------------
   Abstract data type for the P-port and P-Bus opcodes.
-----------------------------------------------------------------------------%

let pbmop =
   define_type 'pbmop'
               'pbmop = PBM_WriteLM | PBM_WritePIU | PBM_WriteCB | PBM_ReadLM |
                        PBM_ReadPIU | PBM_ReadCB | PBM_Illegal';;

let pbsop =
   define_type 'pbsop'
               'pbsop = PBS_Ready | PBS_Illegal';;
```

```
let ibsop =
   define_type 'ibsop'
               'ibsop = IBS_Ready | IBS_Illegal';;

let ibamop =
   define_type 'ibamop'
               'ibamop = IBAM_Ready | IBAM_Illegal';;

let ibasop =
   define_type 'ibasop'
               'ibasop = IBAS_Ready | IBAS_Illegal';;

let rmop =
   define_type 'rmop'
               'rmop = RM_NoReset | RM_Illegal';;

%-----------------------------------------------------------------------------
   Abstract data type for the memory access target.
-----------------------------------------------------------------------------%

let targ_Axiom =
   define_type 'targ_Axiom'
               'targ = LM | PIU | CB';;

%-----------------------------------------------------------------------------
   Abstract data type for the state.
-----------------------------------------------------------------------------%

let pt_state =
   define_type 'pt_state'
               'pt_state = PTState pfsm_ty bool bool';;

let PT_fsm_stateS = new_recursive_definition
    false
    pt_state
    'PT_fsm_stateS'
    "PT_fsm_stateS (PTState PT_fsm_state PT_rqt PT_lock_) = PT_fsm_state";;

let PT_rqtS = new_recursive_definition
    false
    pt_state
    'PT_rqtS'
    "PT_rqtS (PTState PT_fsm_state PT_rqt PT_lock_) = PT_rqt";;

let PT_lock_S = new_recursive_definition
    false
    pt_state
    'PT_lock_S'
    "PT_lock_S (PTState PT_fsm_state PT_rqt PT_lock_) = PT_lock_";;

let State_CASES =
   prove_cases_thm (prove_induction_thm pt_state);;

let PTState_Selectors_Work = prove_thm
   ('PTState_Selectors_Work',
    "!s:pt_state.
      s = (PTState (PT_fsm_stateS s) (PT_rqtS s) (PT_lock_S s))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "s:pt_state" State_CASES)
    THEN REWRITE_TAC [PT_fsm_stateS;PT_rqtS;PT_lock_S]
   );;

%-----------------------------------------------------------------------------
   Abstract data type for the environment.
-----------------------------------------------------------------------------%

let pt_env =
   define_type 'pt_env'
               'pt_env = PTEnv pbmop wordn wordnn wordn wordnn bool ibsop wordnn
                               ibasop rmop';;

let PB_Opcode_inE = new_recursive_definition
```

```
    false
    pt_env
    'PB_Opcode_inE'
    "PB_Opcode_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = PB_Opcode_in";;

let PB_Addr_inE = new_recursive_definition
    false
    pt_env
    'PB_Addr_inE'
    "PB_Addr_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = PB_Addr_in";;

let PB_Data_inE = new_recursive_definition
    false
    pt_env
    'PB_Data_inE'
    "PB_Data_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = PB_Data_in";;

let PB_BS_inE = new_recursive_definition
    false
    pt_env
    'PB_BS_inE'
    "PB_BS_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = PB_BS_in";;

let PB_BE_inE = new_recursive_definition
    false
    pt_env
    'PB_BE_inE'
    "PB_BE_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = PB_BE_in";;

let PB_Lock_inE = new_recursive_definition
    false
    pt_env
    'PB_Lock_inE'
    "PB_Lock_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = PB_Lock_in";;

let IB_Opcode_inE = new_recursive_definition
    false
    pt_env
    'IB_Opcode_inE'
    "IB_Opcode_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = IB_Opcode_in";;

let IB_Data_inE = new_recursive_definition
    false
    pt_env
    'IB_Data_inE'
    "IB_Data_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = IB_Data_in";;

let IBA_Opcode_inE = new_recursive_definition
```

```
    false
    pt_env
    'IBA_Opcode_inE'
    "IBA_Opcode_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = IBA_Opcode_in";;

let Rst_Opcode_inE = new_recursive_definition
    false
    pt_env
    'Rst_Opcode_inE'
    "Rst_Opcode_inE (PTEnv PB_Opcode_in PB_Addr_in PB_Data_in PB_BS_in
                           PB_BE_in PB_Lock_in IB_Opcode_in IB_Data_in
                           IBA_Opcode_in Rst_Opcode_in)
     = Rst_Opcode_in";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm pt_env);;

let PTEnv_Selectors_Work = prove_thm
   ('PTEnv_Selectors_Work',
    "!e:pt_env.
     e = (PTEnv (PB_Opcode_inE e) (PB_Addr_inE e) (PB_Data_inE e) (PB_BS_inE e)
                (PB_BE_inE e) (PB_Lock_inE e) (IB_Opcode_inE e)
                (IB_Data_inE e) (IBA_Opcode_inE e) (Rst_Opcode_inE e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:pt_env" Env_CASES)
    THEN REWRITE_TAC [PB_Opcode_inE; PB_Addr_inE; PB_Data_inE; PB_BS_inE;
                      PB_BE_inE; PB_Lock_inE; IB_Opcode_inE; IB_Data_inE;
                      IBA_Opcode_inE; Rst_Opcode_inE]
   );;

%----------------------------------------------------------------------------
   Abstract data type for the output.
----------------------------------------------------------------------------%

let pt_out =
   define_type 'pt_out'
               'pt_out = PTOut pbmop wordn wordnn wordn wordnn bool
                               ibamop
                               pbsop wordnn';;

let IB_Opcode_outO = new_recursive_definition
    false
    pt_out
    'IB_Opcode_outO'
    "IB_Opcode_outO (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                           IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                           PB_Data_out)
     = IB_Opcode_out";;

let IB_Addr_outO = new_recursive_definition
    false
    pt_out
    'IB_Addr_outO'
    "IB_Addr_outO (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                           IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                           PB_Data_out)
     = IB_Addr_out";;

let IB_Data_outO = new_recursive_definition
    false
    pt_out
    'IB_Data_outO'
    "IB_Data_outO (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                           IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                           PB_Data_out)
     = IB_Data_out";;

let IB_BS_outO = new_recursive_definition
    false
```

```
    pt_out
    'IB_BS_out0'
    "IB_BS_out0 (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                       IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                       PB_Data_out)
     = IB_BS_out";;

let IB_BE_out0 = new_recursive_definition
    false
    pt_out
    'IB_BE_out0'
    "IB_BE_out0 (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                       IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                       PB_Data_out)
     = IB_BE_out";;

let IB_Lock_out0 = new_recursive_definition
    false
    pt_out
    'IB_Lock_out0'
    "IB_Lock_out0 (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                       IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                       PB_Data_out)
     = IB_Lock_out";;

let IBA_Opcode_out0 = new_recursive_definition
    false
    pt_out
    'IBA_Opcode_out0'
    "IBA_Opcode_out0 (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                       IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                       PB_Data_out)
     = IBA_Opcode_out";;

let PB_Opcode_out0 = new_recursive_definition
    false
    pt_out
    'PB_Opcode_out0'
    "PB_Opcode_out0 (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                       IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                       PB_Data_out)
     = PB_Opcode_out";;

let PB_Data_out0 = new_recursive_definition
    false
    pt_out
    'PB_Data_out0'
    "PB_Data_out0 (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out
                       IB_BE_out IB_Lock_out IBA_Opcode_out PB_Opcode_out
                       PB_Data_out)
     = PB_Data_out";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm pt_out);;

let PTOut_Selectors_Work = prove_thm
   ('PTOut_Selectors_Work',
    "!p:pt_out.
     p = (PTOut (IB_Opcode_out0 p) (IB_Addr_out0 p) (IB_Data_out0 p)
                (IB_BS_out0 p) (IB_BE_out0 p) (IB_Lock_out0 p)
                (IBA_Opcode_out0 p) (PB_Opcode_out0 p) (PB_Data_out0 p))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "p:pt_out" Out_CASES)
    THEN REWRITE_TAC [IB_Opcode_out0; IB_Addr_out0; IB_Data_out0; IB_BS_out0;
                      IB_BE_out0; IB_Lock_out0; PB_Opcode_out0; PB_Data_out0;
                      IBA_Opcode_out0]
   );;

close_theory();;

%----------------------------------------------------------------------------
```

```
   File:      ptransp_def.ml

   Author:    (c) D.A. Fura 1992-93

   Date:      2 March 1993

   This file contains the ml source for the trans-level specification of the
   P-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.

-------------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/pport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing',true);;

system 'rm ptransp_def.th';;

new_theory 'ptransp_def';;

map new_parent ['ptauxp_def';'array_def';'wordn_def'];;

%-------------------------------------------------------------------------------
   Next state definition for P-Port instructions.
-------------------------------------------------------------------------------%

let PT_WriteNSF_A = new_definition
   ('PT_WriteNSF_A',
    "! (s :pt_state) (e :pt_env) .
     PT_WriteNSF_A s e =
        let new_PT_fsm_state = PA in
        let new_PT_rqt = F in
        let new_PT_lock_ = PB_Lock_inE e in
        (PTState new_PT_fsm_state new_PT_rqt new_PT_lock_)"
   );;

let PT_WriteNSF_H = new_definition
   ('PT_WriteNSF_H',
    "! (s :pt_state) (e :pt_env) .
     PT_WriteNSF_H s e =
        let new_PT_fsm_state = PH in
        let new_PT_rqt = F in
        let new_PT_lock_ = PB_Lock_inE e in
        (PTState new_PT_fsm_state new_PT_rqt new_PT_lock_)"
   );;

let PT_ReadNSF_A = new_definition
   ('PT_ReadNSF_A',
    "! (s :pt_state) (e :pt_env) .
     PT_ReadNSF_A s e =
        let new_PT_fsm_state = PA in
        let new_PT_rqt = F in
        let new_PT_lock_ = PB_Lock_inE e in
        (PTState new_PT_fsm_state new_PT_rqt new_PT_lock_)"
   );;

let PT_ReadNSF_H = new_definition
   ('PT_ReadNSF_H',
    "! (s :pt_state) (e :pt_env) .
     PT_ReadNSF_H s e =
        let new_PT_fsm_state = PH in
        let new_PT_rqt = F in
        let new_PT_lock_ = PB_Lock_inE e in
        (PTState new_PT_fsm_state new_PT_rqt new_PT_lock_)"
   );;

%-------------------------------------------------------------------------------
   Output definition for P-Port instructions.
```

```
------------------------------------------------------------------------------%

let PT_WriteOF = new_definition
   ('PT_WriteOF',
    "! (s :pt_state) (e :pt_env) .
     PT_WriteOF s e =
        let IB_Opcode_out = PB_Opcode_inE e in
        let IB_Addr_out = PB_Addr_inE e in
        let IB_Data_out = PB_Data_inE e in
        let IB_BS_out = PB_BS_inE e in
        let IB_BE_out = PB_BE_inE e in
        let IB_Lock_out = PB_Lock_inE e in
        let IBA_Opcode_out = IBAM_Ready in
        let PB_Opcode_out = PBS_Ready in
        let PB_Data_out = ARBN in

        (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out IB_BE_out
               IB_Lock_out IBA_Opcode_out PB_Opcode_out PB_Data_out)"
   );;

let PT_ReadOF = new_definition
   ('PT_ReadOF',
    "! (s :pt_state) (e :pt_env) .
     PT_ReadOF s e =
        let IB_Opcode_out = PB_Opcode_inE e in
        let IB_Addr_out = PB_Addr_inE e in
        let IB_Data_out = ARBN in
        let IB_BS_out = PB_BS_inE e in
        let IB_BE_out = PB_BE_inE e in
        let IB_Lock_out = PB_Lock_inE e in
        let IBA_Opcode_out = IBAM_Ready in
        let PB_Opcode_out = PBS_Ready in
        let PB_Data_out = IB_Data_inE e in

        (PTOut IB_Opcode_out IB_Addr_out IB_Data_out IB_BS_out IB_BE_out
               IB_Lock_out IBA_Opcode_out PB_Opcode_out PB_Data_out)"
   );;

%-----------------------------------------------------------------------------
   P-Port interpreter definition.
------------------------------------------------------------------------------%

let PT_Exec = new_definition
   ('PT_Exec',
    "! (pti :PTI) (s :timeT->pt_state) (e :timeT->pt_env) (p :timeT->pt_out)
       (t :timeT) .
     PT_Exec pti s e p t =
        (Rst_Opcode_inE (e t) = RM_NoReset) /\
        (IB_Opcode_inE (e t) = IBS_Ready) /\
        (IBA_Opcode_inE (e t) = IBAS_Ready) /\
        ((pti = PT_Write) =>
            ((PB_Opcode_inE (e t) = PBM_WriteLM) \/
             (PB_Opcode_inE (e t) = PBM_WritePIU) \/
             (PB_Opcode_inE (e t) = PBM_WriteCB))
      % (pti = PT_Read) % |
            ((PB_Opcode_inE (e t) = PBM_ReadLM) \/
             (PB_Opcode_inE (e t) = PBM_ReadPIU) \/
             (PB_Opcode_inE (e t) = PBM_ReadCB)))"
   );;

let PT_PreC = new_prim_rec_definition
   ('PT_PreC',
    "(PT_PreC (pti:PTI) s e p 0 =
        ~(PT_fsm_stateS (s 0) = PD) /\
        (PT_rqtS (s 0) = F)) /\
     (PT_PreC pti s e p (SUC t) =
        ~(PT_fsm_stateS (s (SUC t)) = PD) /\
        (PT_rqtS (s (SUC t)) = F) /\
        ((PT_Exec PT_Write s e p t /\ PT_PreC PT_Write s e p t) \/
         (PT_Exec PT_Read s e p t /\ PT_PreC PT_Read s e p t)))"
   );;
```

```
let PT_PostC = new_definition
   ('PT_PostC',
    "! (pti :PTI) (s :timeT->pt_state) (e :timeT->pt_env) (p :timeT->pt_out)
       (t :timeT) .
     PT_PostC pti s e p t =
        (pti = PT_Write) => (((s (t+1) = PT_WriteNSF_A (s t) (e t)) \/
                             (s (t+1) = PT_WriteNSF_H (s t) (e t))) /\
                            (p t = PT_WriteOF (s t) (e t)))
      % (pti = PT_Read) % | (((s (t+1) = PT_ReadNSF_A (s t) (e t)) \/
                             (s (t+1) = PT_ReadNSF_H (s t) (e t))) /\
                            (p t = PT_ReadOF (s t) (e t)))"
   );;

let PT_Correct = new_definition
   ('PT_Correct',
    "! (pti :PTI) (s :timeT->pt_state) (e :timeT->pt_env) (p :timeT->pt_out)
       (t :timeT) .
     PT_Correct pti s e p t =
        PT_Exec pti s e p t /\
        PT_PreC pti s e p t
        ==>
        PT_PostC pti s e p t"
   );;

let PTSet_Correct = new_definition
   ('PTSet_Correct',
    "! (s :timeT->pt_state) (e :timeT->pt_env) (p :timeT->pt_out) .
     PTSet_Correct s e p = !(pti:PTI)(t:timeT) . PT_Correct pti s e p t"
   );;

close_theory();;


%----------------------------------------------------------------------------

   File:      ptabs_def.ml

   Author:    (c) D.A. Fura 1992-93

   Date:      6 March 1993

----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/pport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

set_flag ('timing',true);;

system 'rm ptabs_def.th';;

new_theory 'ptabs_def';;

map new_parent ['piuaux_def';'ptauxp_def';'paux_def';'array_def';'wordn_def';
                'busn_def';'templogic_def';'ptransp_def';'pclock_def'];;

new_type_abbrev ('time',":num");;
new_type_abbrev ('timeT',":num");;
new_type_abbrev ('timeC',":num");;

% L_Bus start-of-transaction signal. %
let ale_sig_pb = new_definition
    ('ale_sig_pb',
     "! (e' :timeC->pc_env) .
      ale_sig_pb e' = \u':timeC. ~BSel(L_ads_E (e' u')) /\ BSel(L_den_E (e' u'))"
    );;

% I_Bus start-of-transaction signal for the P Process. %
let ale_sig_ib = new_definition
    ('ale_sig_ib',
```

```
    "! (p' :timeC->pc_out) .
     ale_sig_ib p' = \u':timeC. BSel(I_hlda_O (p' u')) /\
                                ((BSel(I_male_O (p' u')) = LO) \/
                                 (BSel(I_rale_O (p' u')) = LO) \/
                                 (BSel(I_cale_O (p' u')) = F))"
   );;

% I_Bus end-of-transaction signal (I-Bus master perspective). %
let ack_sig_ib = new_definition
   ('ack_sig_ib',
    "! (e' :timeC->pc_env) (p' :timeC->pc_out) .
     ack_sig_ib e' p' =
        \u':timeC. (BSel(I_last_O (p' u')) = LO) /\ ~BSel(I_srdy_E (e' u'))"
   );;

% I_Bus end-of-data-word signal - transaction not finished. %
let rdy_sig_ib = new_definition
   ('rdy_sig_ib',
    "! (e' :timeC->pc_env) (p' :timeC->pc_out) .
     rdy_sig_ib e' p' =
        \u':timeC. (BSel(I_last_O (p' u')) = HI) /\ ~BSel(I_srdy_E (e' u'))"
   );;

%-----------------------------------------------------------------------------
   Abstraction predicate for an Intel 80960 L_Bus slave.
-----------------------------------------------------------------------------%

let PB_Slave = new_definition
   ('PB_Slave',
    "PB_Slave (pti :PTI)
              (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
              (e' :timeC->pc_env) (p' :timeC->pc_out) (tp' :timeC) =

       % slave-ready 0,1,2,3 times. %
       let t'rdy0 = @u'. NTH_TIME_FALSE 0 (bsig L_ready_O p') tp' u' in
       let t'rdy1 = @u'. NTH_TIME_FALSE 1 (bsig L_ready_O p') tp' u' in
       let t'rdy2 = @u'. NTH_TIME_FALSE 2 (bsig L_ready_O p') tp' u' in
       let t'rdy3 = @u'. NTH_TIME_FALSE 3 (bsig L_ready_O p') tp' u' in
       let write = (ASel(L_wrE (e' tp'))) in
       let read = (~write) in
       let bs = (VAL 1 (SUBARRAY (BSel(L_ad_inE (e' tp'))) (1,0))) in
       % end-of-transaction time. %
       let t'ack = @u'. NTH_TIME_FALSE bs (bsig L_ready_O p') tp' u' in
       let valid_rqt =
           (!u'. LESS_THAN_N_TIMES_FALSE bs (bsig L_ready_O p') tp' u' ==>
                    STABLE_FALSE (ale_sig_pb e') (tp'+1,u'+1)) in
       let valid_ack =
           ((?t'ack. N_TIMES_FALSE bs (bsig L_ready_O p') tp' t'ack) /\
            (STABLE_AB_OFFn (sig L_ad_outO p') (tp',tp')) /\
            (write ==>
              (!u'. STABLE_FALSE (ale_sig_pb e') (tp'+1,u') ==>
                       STABLE_AB_OFFn (sig L_ad_outO p') (tp'+1,u'))) /\
            (!u'. STABLE_FALSE (ale_sig_pb e') (t'ack,u') ==>
                     STABLE_AB_OFFn (sig L_ad_outO p') (t'ack+1,u'))) in
       let lmem =
           ((ELEMENT (ASel(L_ad_inE (e' tp'))) (31) = F) /\
            (~(SUBARRAY (ASel(L_ad_inE (e' tp'))) (25,24) = (WORDN 1 3)))) in
       let piu =
           ((ELEMENT (ASel(L_ad_inE (e' tp'))) (31) = F) /\
            (SUBARRAY (ASel(L_ad_inE (e' tp'))) (25,24) = (WORDN 1 3))) in
       let cbus =
           (ELEMENT (ASel(L_ad_inE (e' tp'))) (31) = T) in
       let dw0 =
           ((write /\ STABLE_AB (sig L_ad_inE e') (tp'+1,t'rdy0))
            => (ASel(L_ad_inE (e' t'rdy0))) | ARBN) in
       let dw1 =
           ((write /\ (bs > 0) /\ STABLE_AB(sig L_ad_inE e')(t'rdy0+1,t'rdy1))
            => (ASel(L_ad_inE (e' t'rdy1))) | ARBN) in
       let dw2 =
           ((write /\ (bs > 1) /\ STABLE_AB(sig L_ad_inE e')(t'rdy1+1,t'rdy2))
            => (ASel(L_ad_inE (e' t'rdy2))) | ARBN) in
       let dw3 =
```

```
          ((write /\ (bs > 2) /\ STABLE_AB(sig L_ad_inE e')(t'rdy2+1,t'rdy3))
           => (ASel(L_ad_inE (e' t'rdy3))) | ARBN) in
       let be0 = (ASel(L_be_E (e' tp'))) in
       let be1 =
             (((bs > 0) /\ STABLE_AB(sig L_be_E e')(t'rdy0+1,t'rdy1))
              => (ASel(L_be_E (e' t'rdy1))) | ARBN) in
       let be2 =
             (((bs > 1) /\ STABLE_AB(sig L_be_E e')(t'rdy1+1,t'rdy2))
              => (ASel(L_be_E (e' t'rdy2))) | ARBN) in
       let be3 =
             (((bs > 2) /\ STABLE_AB(sig L_be_E e')(t'rdy2+1,t'rdy3))
              => (ASel(L_be_E (e' t'rdy3))) | ARBN) in
       let dr0 = (wordnVAL (BSel(L_ad_outO (p' t'rdy0)))) in
       let dr1 = ((bs > 0) => wordnVAL (BSel(L_ad_outO (p' t'rdy1))) | ARBN) in
       let dr2 = ((bs > 1) => wordnVAL (BSel(L_ad_outO (p' t'rdy2))) | ARBN) in
       let dr3 = ((bs > 2) => wordnVAL (BSel(L_ad_outO (p' t'rdy3))) | ARBN) in

       ((PB_Opcode_inE (e t) =
          valid_rqt =>
           (lmem => (write => PBM_WriteLM | PBM_ReadLM) |
            piu => (write => PBM_WritePIU | PBM_ReadPIU) |
            cbus => (write => PBM_WriteCB | PBM_ReadCB) | PBM_Illegal) |
            PBM_Illegal) /\
        (PB_Addr_inE (e t) = SUBARRAY (ASel(L_ad_inE (e' tp'))) (25,2)) /\
        (PB_Data_inE (e t) =
           ALTER (ALTER (ALTER (ALTER ARBN 0 dw0) 1 dw1) 2 dw2) 3 dw3) /\
        (PB_BS_inE (e t) = SUBARRAY (BSel(L_ad_inE (e' tp'))) (1,0)) /\
        (PB_BE_inE (e t) =
           ALTER (ALTER (ALTER (ALTER ARBN 0 be0) 1 be1) 2 be2) 3 be3) /\
        (PB_Lock_inE (e t) = BSel(L_lock_E (e' tp'))) /\

        (PB_Opcode_outO (p t) = (valid_ack => PBS_Ready | PBS_Illegal)) /\
        (PB_Data_outO (p t) =
           ALTER (ALTER (ALTER (ALTER ARBN 0 dr0) 1 dr1) 2 dr2) 3 dr3))"
   );;

%-----------------------------------------------------------------------------
   Abstraction predicate for an I-Bus arbitration master.
-----------------------------------------------------------------------------%

let IBA_PMaster = new_definition
   ('IBA_PMaster',
    "IBA_PMaster (pti :PTI)
                 (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
                 (e' :timeC->pc_env) (p' :timeC->pc_out) =
       (IBA_Opcode_inE (e t) =
          ((!u'. ?v'. STABLE_FALSE_THEN_TRUE (bsig I_hold_E e') (u',v')) /\
           (!u'. CHANGES_FALSE (bsig I_crqt_O p') u' ==>
                   (?v'. (u' < v') /\
                         STABLE_TRUE_THEN_FALSE (bsig I_cgnt_E e') (u',v'))) /\
           (!u'. BSel(I_crqt_O (p' u')) ==> BSel(I_cgnt_E (e' u'))) /\
           (!u'. ~BSel(I_cgnt_E (e' u')) ==>
                    (BSel(I_hold_E (e' u')) /\ BSel(I_hold_E (e' (u'-1))))))
          => IBAS_Ready | IBAS_Illegal) /\

       (IBA_Opcode_outO (p t) =
          ((!u'. ?v'. STABLE_FALSE_THEN_TRUE (bsig I_hlda_O p') (u',v')) /\
           (!u'. CHANGES_FALSE (bsig I_hold_E e') u' ==>
                   (?v'. (u' < v') /\
                         STABLE_TRUE_THEN_FALSE (bsig I_hlda_O p') (u',v'))))
          => IBAM_Ready | IBAM_Illegal)"
   );;

%-----------------------------------------------------------------------------
   Abstraction predicate for an I-Bus master.
-----------------------------------------------------------------------------%

let IB_PMaster = new_definition
   ('IB_PMaster',
    "IB_PMaster (pti :PTI)
                (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
                (e' :timeC->pc_env) (p' :timeC->pc_out) (ti' :timeC) =
```

```
let write = (ELEMENT (BSel(I_ad_out0 (p' ti'))) (27) = HI) in
let read = (ELEMENT (BSel(I_ad_out0 (p' ti'))) (27) = LO) in
let bs =
   (VAL 1 (SUBARRAY (wordnVAL (BSel(I_ad_out0 (p' ti')))) (25,24))) in
let lmem = (BSel(I_male_O (p' ti')) = LO) in
let piu = (BSel(I_rale_O (p' ti')) = LO) in
let cbus = (BSel(I_cale_O (p' ti')) = F) in
let valid_rqt = % I-Bus master control signals valid %
    ((lmem ==> (~piu /\ ~cbus)) /\
     (piu ==> (~cbus)) /\
     (?u'. STABLE_HI_THEN_LO (bsig I_last_O p') (ti',u'))) in
let valid_ack = % I-Bus slave control signals valid %
    ((?u'. STABLE_TRUE_THEN_FALSE (bsig I_srdy_E e') (ti'+1,u')) /\
     (!u'. rdy_sig_ib e' p' u' ==>
           (?v'. STABLE_TRUE_THEN_FALSE (bsig I_srdy_E e') (u'+1,v')))) in
% end-of-active-transaction time %
let t'ack = @u'. STABLE_FALSE_THEN_TRUE (ack_sig_ib e' p') (ti',u') in
% slave_ready 0,1,2,3 times %
let t'rdy0 = @u'. NTH_TIME_FALSE 0 (bsig I_srdy_E e') (ti'+1) u' in
let t'rdy1 = @u'. NTH_TIME_FALSE 1 (bsig I_srdy_E e') (ti'+1) u' in
let t'rdy2 = @u'. NTH_TIME_FALSE 2 (bsig I_srdy_E e') (ti'+1) u' in
let t'rdy3 = @u'. NTH_TIME_FALSE 3 (bsig I_srdy_E e') (ti'+1) u' in
% data-valid predicates for write-data words 0,1,2,3 %
let dv0 = (t'rdy0 <= t'ack /\
           STABLE_AB (sig I_ad_out0 p') (ti'+1,t'rdy0)) in
let dv1 = (t'rdy1 <= t'ack /\
           STABLE_AB (sig I_ad_out0 p') (t'rdy0+1,t'rdy1)) in
let dv2 = (t'rdy2 <= t'ack /\
           STABLE_AB (sig I_ad_out0 p') (t'rdy1+1,t'rdy2)) in
let dv3 = (t'rdy3 <= t'ack /\
           STABLE_AB (sig I_ad_out0 p') (t'rdy2+1,t'rdy3)) in
% write-data words 0,1,2,3 %
let d0 = (dv0 => wordnVAL (BSel(I_ad_out0 (p' t'rdy0))) | ARBN) in
let d1 = (dv1 => wordnVAL (BSel(I_ad_out0 (p' t'rdy1))) | ARBN) in
let d2 = (dv2 => wordnVAL (BSel(I_ad_out0 (p' t'rdy2))) | ARBN) in
let d3 = (dv3 => wordnVAL (BSel(I_ad_out0 (p' t'rdy3))) | ARBN) in
% byte-enable-valid predicates for byte enables 1,2,3 %
let bv1 = (t'rdy0 <= t'ack /\
           STABLE_AB (sig I_be_O p') (ti'+1,t'rdy0)) in
let bv2 = (t'rdy1 <= t'ack /\
           STABLE_AB (sig I_be_O p') (t'rdy0+1,t'rdy1)) in
let bv3 = (t'rdy2 <= t'ack /\
           STABLE_AB (sig I_be_O p') (t'rdy1+1,t'rdy2)) in
% byte enables 0,1,2,3 %
let b0 = NOTN 3 (wordnVAL (BSel(I_be_O (p' ti')))) in
let b1 = (bv1 => NOTN 3 (wordnVAL (BSel(I_be_O (p' t'rdy0)))) | ARBN) in
let b2 = (bv2 => NOTN 3 (wordnVAL (BSel(I_be_O (p' t'rdy1)))) | ARBN) in
let b3 = (bv3 => NOTN 3 (wordnVAL (BSel(I_be_O (p' t'rdy2)))) | ARBN) in
% data-valid predicates for read-data words 0,1,2,3 %
let ev0 = (t'rdy0 <= t'ack) in
let ev1 = (t'rdy1 <= t'ack) in
let ev2 = (t'rdy2 <= t'ack) in
let ev3 = (t'rdy3 <= t'ack) in
% read-data words 0,1,2,3 %
let e0 = (ev0 => BSel(I_ad_inE (e' t'rdy0)) | ARBN) in
let e1 = (ev1 => BSel(I_ad_inE (e' t'rdy1)) | ARBN) in
let e2 = (ev2 => BSel(I_ad_inE (e' t'rdy2)) | ARBN) in
let e3 = (ev3 => BSel(I_ad_inE (e' t'rdy3)) | ARBN) in

((IB_Opcode_out0 (p t) =
    (valid_rqt /\ write /\ lmem) => PBM_WriteLM |
    (valid_rqt /\ write /\ piu) => PBM_WritePIU |
    (valid_rqt /\ write /\ cbus) => PBM_WriteCB |
    (valid_rqt /\ read /\ lmem) => PBM_ReadLM |
    (valid_rqt /\ read /\ piu) => PBM_ReadPIU |
    (valid_rqt /\ read /\ cbus) => PBM_ReadCB | PBM_Illegal) /\
 (IB_Addr_out0 (p t) =
    SUBARRAY (wordnVAL (BSel(I_ad_out0 (p' ti')))) (23,0)) /\
 (IB_Data_out0 (p t) =
    ALTER (ALTER (ALTER (ALTER ARBN 0 d0) 1 d1) 2 d2) 3 d3) /\
 (IB_BS_out0 (p t) =
```

```
        (STABLE_LO (bsig I_last_O p') (ti'+1,t'rdy0)) => WORDN 1 0 |
        (STABLE_HI (bsig I_last_O p') (ti'+1,t'rdy0) /\
         STABLE_LO (bsig I_last_O p') (t'rdy0+1,t'rdy1)) => WORDN 1 1 |
        (STABLE_HI (bsig I_last_O p') (ti'+1,t'rdy1) /\
         STABLE_LO (bsig I_last_O p') (t'rdy1+1,t'rdy2)) => WORDN 1 2 |
        (STABLE_HI (bsig I_last_O p') (ti'+1,t'rdy2) /\
         STABLE_LO (bsig I_last_O p') (t'rdy2+1,t'rdy3)) => WORDN 1 3 | ARBN) /\
       (IB_BE_outO (p t) =
         ALTER (ALTER (ALTER (ALTER ARBN 0 b0) 1 b1) 2 b2) 3 b3) /\

       (IB_Opcode_inE (e t) = valid_ack => IBS_Ready | IBS_Illegal) /\
       (IB_Data_inE (e t) =
         ALTER (ALTER (ALTER (ALTER ARBN 0 e0) 1 e1) 2 e2) 3 e3))"
  );;

let Rst_Slave = new_definition
   ('Rst_Slave',
    "Rst_Slave (pti :PTI) (e :timeT->pt_env) (t :timeT) (e' :timeC->pc_env) =
        Rst_Opcode_inE (e t) =
           (!u':timeC. BSel(RstE (e' u')) = F) => RM_NoReset | RM_Illegal"
   );;

let PStateAbs = new_definition
   ('PStateAbs',
    "PStateAbs (pti :PTI) (s :timeT->pt_state) (e :timeT->pt_env)
               (p :timeT->pt_out) (t :timeT) (s' :timeC->pc_state)
               (e' :timeC->pc_env) (p' :timeC->pc_out) (tp' :timeC) =

      % (t' = 0) %
       (~(P_fsm_stateS (s' 0) = PD) /\
        (P_rqtS (s' 0) = F) /\
        (P_lock_S (s' 0) = T)) /\
       ((tp' > 0) ==>
          (P_fsm_stateS (s' tp') = PT_fsm_stateS (s t)) /\
          (P_rqtS (s' tp') = PT_rqtS (s t)) /\
          (P_lock_S (s' tp') = PT_lock_S (s t))) /\
       (!tp'suc:timeC.
        NTH_TIME_TRUE (SUC t) (ale_sig_pb e') 1 tp'suc ==>
          (PT_fsm_stateS (s (t+1)) = P_fsm_stateS (s' tp'suc)) /\
          (PT_rqtS (s (t+1)) = P_rqtS (s' tp'suc)) /\
          (PT_lock_S (s (t+1)) = P_lock_S (s' tp'suc)))"
   );;

let PTAbs = new_definition
   ('PTAbs',
    "PTAbs (pti :PTI) (s :timeT->pt_state) (e :timeT->pt_env)
           (p :timeT->pt_out) (t :timeT) (s' :timeC->pc_state)
           (e' :timeC->pc_env) (p' :timeC->pc_out) =

       (PT_Exec pti s e p t
        ==> ?tp'. NTH_TIME_TRUE t (ale_sig_pb e') 0 tp' /\ (tp' > 0)) /\
       (!tp'. NTH_TIME_TRUE t (ale_sig_pb e') 0 tp'
              ==> (Rst_Slave pti e t e' /\
                   PB_Slave pti e p t e' p' tp' /\
                   IBA_PMaster pti e p t e' p' /\
                   PStateAbs pti s e p t s' e' p' tp')) /\
       (!ti'. NTH_TIME_TRUE t (ale_sig_ib p') 0 ti'
              ==> IB_PMaster pti e p t e' p' ti')"
   );;

let PTAbsSet = new_definition
   ('PTAbsSet',
    "PTAbsSet (s :timeT->pt_state) (e :timeT->pt_env) (p :timeT->pt_out)
              (s' :timeC->pc_state) (e' :timeC->pc_env) (p' :timeC->pc_out) =
       !(pti:PTI)(t:timeT). PTAbs pti s e p t s' e' p'"
   );;

loadf 'aux_defs';;
let ASel = definition 'piuaux_def' 'ASel';;
let BSel = definition 'piuaux_def' 'BSel';;

let PB_Addr_in_ISO = prove_thm
```

```
   ('PB_Addr_in_ISO',
    "! (pti :PTI) (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
       (e' :timeC->pc_env) (p' :timeC->pc_out) (t' :timeC) .
     PB_Slave pti e p t e' p' t' ==>
        (PB_Addr_inE (e t) = SUBARRAY (FST(L_ad_inE (e' t'))) (25,2))",
    REWRITE_TAC [EXPAND_LET_RULE PB_Slave;ASel]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
   );;

let PB_BS_in_ISO = prove_thm
   ('PB_BS_in_ISO',
    "! (pti :PTI) (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
       (e' :timeC->pc_env) (p' :timeC->pc_out) (t' :timeC) .
     PB_Slave pti e p t e' p' t' ==>
        (PB_BS_inE (e t) = SUBARRAY (SND(L_ad_inE (e' t'))) (1,0))",
    REWRITE_TAC [EXPAND_LET_RULE PB_Slave;BSel]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
   );;

let RM_Opcode_in_ISO = prove_thm
   ('RM_Opcode_in_ISO',
    "! (pti :PTI) (e :timeT->pt_env) (t :timeT) (e' :timeC->pc_env) .
     Rst_Slave pti e t e' ==>
        (Rst_Opcode_inE (e t) =
           (!u':timeC. BSel(RstE (e' u')) = F) => RM_NoReset | RM_Illegal)",
    REWRITE_TAC [Rst_Slave;BSel]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
   );;

let IB_Opcode_in_ISO = prove_thm
   ('IB_Opcode_in_ISO',
    "! (pti :PTI) (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
       (e' :timeC->pc_env) (p' :timeC->pc_out) (ti' :timeC) .
     IB_PMaster pti e p t e' p' ti' ==>
       (IB_Opcode_inE (e t) =
           ((?u'. STABLE_TRUE_THEN_FALSE (bsig I_srdy_E e') (ti'+1,u')) /\
            (!u'. rdy_sig_ib e' p' u' ==>
                  (?v'. STABLE_TRUE_THEN_FALSE (bsig I_srdy_E e') (u'+1,v'))))
            => IBS_Ready | IBS_Illegal)",
    REWRITE_TAC [EXPAND_LET_RULE IB_PMaster;BSel]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
   );;

let IBA_Opcode_in_ISO = prove_thm
   ('IBA_Opcode_in_ISO',
    "! (pti :PTI) (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
       (e' :timeC->pc_env) (p' :timeC->pc_out) (t' :timeC) .
     IBA_PMaster pti e p t e' p' ==>
        (IBA_Opcode_inE (e t) =
           ((!u'. ?v'. STABLE_FALSE_THEN_TRUE (bsig I_hold_E e') (u',v')) /\
            (!u'. CHANGES_FALSE (bsig I_crqt_O p') u' ==>
                    (?v'. (u' < v') /\
                          STABLE_TRUE_THEN_FALSE (bsig I_cgnt_E e') (u',v'))) /\
            (!u'. BSel(I_crqt_O (p' u')) ==> BSel(I_cgnt_E (e' u'))) /\
            (!u'. ~BSel(I_cgnt_E (e' u')) ==>
                    (BSel(I_hold_E (e' u')) /\ BSel(I_hold_E (e' (u'-1))))))
          => IBAS_Ready | IBAS_Illegal)",
    REWRITE_TAC [EXPAND_LET_RULE IBA_PMaster;BSel]
    THEN REPEAT STRIP_TAC
    THEN ASM_REWRITE_TAC[]
   );;

let IB_Addr_out_ISO = prove_thm
   ('IB_Addr_out_ISO',
    "! (pti :PTI) (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
       (e' :timeC->pc_env) (p' :timeC->pc_out) (t' :timeC) .
     IB_PMaster pti e p t e' p' t' ==>
        (IB_Addr_outO (p t) =
```

```
        SUBARRAY (wordnVAL(SND(I_ad_outO (p' t')))) (23,0))",
   REWRITE_TAC [EXPAND_LET_RULE IB_PMaster;BSel]
   THEN REPEAT STRIP_TAC
   THEN ASM_REWRITE_TAC[]
  );;

let IB_BS_out_ISO = prove_thm
  ('IB_BS_out_ISO',
   "! (pti :PTI) (e :timeT->pt_env) (p :timeT->pt_out) (t :timeT)
      (e' :timeC->pc_env) (p' :timeC->pc_out) (t' :timeC) .
    IB_PMaster pti e p t e' p' t' ==>
       (let t'rdy0 = @u'. NTH_TIME_FALSE 0(bsig I_srdy_E e')(t' + 1)u' in
        let t'rdy1 = @u'. NTH_TIME_FALSE 1(bsig I_srdy_E e')(t' + 1)u' in
        let t'rdy2 = @u'. NTH_TIME_FALSE 2(bsig I_srdy_E e')(t' + 1)u' in
        let t'rdy3 = @u'. NTH_TIME_FALSE 3(bsig I_srdy_E e')(t' + 1)u' in
        (IB_BS_outO(p t) =
           (STABLE_LO(bsig I_last_O p')(t' + 1,t'rdy0) => WORDN 1 0 |
            ((STABLE_HI(bsig I_last_O p')(t' + 1,t'rdy0) /\
              STABLE_LO(bsig I_last_O p')(t'rdy0 + 1,t'rdy1)) => WORDN 1 1 |
            ((STABLE_HI(bsig I_last_O p')(t' + 1,t'rdy1) /\
              STABLE_LO(bsig I_last_O p')(t'rdy1 + 1,t'rdy2)) => WORDN 1 2 |
            ((STABLE_HI(bsig I_last_O p')(t' + 1,t'rdy2) /\
              STABLE_LO(bsig I_last_O p')(t'rdy2 + 1,t'rdy3)) => WORDN 1 3 |
            ARBN))))))",
   EXPAND_LET_TAC
   THEN REWRITE_TAC [EXPAND_LET_RULE IB_PMaster;BSel]
   THEN REPEAT STRIP_TAC
   THEN ASM_REWRITE_TAC[]
  );;

close_theory();;
```

4.3 M-Port Transaction-Level Specification

This section contains the theories *mtauxp_def*, *mtransp_def*, and *mtabs_def*, defining the M-Port transaction-level data structures, interpreter, and abstraction, respectively.

```
%-----------------------------------------------------------------------------

   File:      mtauxp_def.ml

   Author:    (c) D.A. Fura 1993

   Date:      2 March 1993

   This file contains types and definitions for the transaction-level
   specification of the P-Process of the PIU M-Port.

-----------------------------------------------------------------------------%

set_flag ('timing',true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/ftep/piu/hol/mport/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm mtauxp_def.th';;

new_theory 'mtauxp_def';;

map new_parent ['maux_def'];;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('timeT', ":num");;
new_type_abbrev ('wordn', ":num->bool");;
```

```
new_type_abbrev ('wordnn', ":num->wordn");;

%-----------------------------------------------------------------------------
   Abstract data type for the M-port instruction set.
-----------------------------------------------------------------------------%

let MTI =
   define_type 'MTI'
               'MTI = MT_Write | MT_Read | MT_Idle';;

%-----------------------------------------------------------------------------
   Abstract data type for the M-Port transaction opcodes.
-----------------------------------------------------------------------------%

% P-Bus Master Opcodes %
let pbmop =
   define_type 'pbmop'
               'pbmop = PBM_WriteLM | PBM_WritePIU | PBM_WriteCB | PBM_ReadLM |
                        PBM_ReadPIU | PBM_ReadCB | PBM_Illegal';;

% I-Bus Slave Opcodes %
let ibsop =
   define_type 'ibsop'
               'ibsop = IBS_Ready | IBS_Idle | IBS_Illegal';;

% I-Bus Arbitration-Master Opcodes %
let ibamop =
   define_type 'ibamop'
               'ibamop = IBAM_ProcP | IBAM_ProcC | IBAM_Illegal';;

% M-Bus Master Opcodes %
let mbmop =
   define_type 'mbmop'
               'mbmop = MBM_WriteLM | MBM_ReadLM | MBM_Idle | MBM_Illegal';;

% M-Bus Slave Opcodes %
let mbsop =
   define_type 'mbsop'
               'mbsop = MBS_Ready | MBS_Illegal';;

% Reset Master Opcodes %
let rmop =
   define_type 'rmop'
               'rmop = RM_NoReset | RM_Illegal';;

%-----------------------------------------------------------------------------
   Abstract data type for the environment.
-----------------------------------------------------------------------------%

let mt_env =
   define_type 'mt_env'
               'mt_env = MTEnv pbmop wordn wordnn wordn wordnn
                               mbsop wordnn
                               ibamop
                               rmop';;

let IB_Opcode_inE = new_recursive_definition
    false
    mt_env
    'IB_Opcode_inE'
    "IB_Opcode_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = IB_Opcode_in";;

let IB_Addr_inE = new_recursive_definition
    false
    mt_env
    'IB_Addr_inE'
    "IB_Addr_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                        MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = IB_Addr_in";;
```

```
let IB_Data_inE = new_recursive_definition
    false
    mt_env
    'IB_Data_inE'
    "IB_Data_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = IB_Data_in";;

let IB_BS_inE = new_recursive_definition
    false
    mt_env
    'IB_BS_inE'
    "IB_BS_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = IB_BS_in";;

let IB_BE_inE = new_recursive_definition
    false
    mt_env
    'IB_BE_inE'
    "IB_BE_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = IB_BE_in";;

let MB_Opcode_inE = new_recursive_definition
    false
    mt_env
    'MB_Opcode_inE'
    "MB_Opcode_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = MB_Opcode_in";;

let MB_Data_inE = new_recursive_definition
    false
    mt_env
    'MB_Data_inE'
    "MB_Data_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = MB_Data_in";;

let IBAM_Opcode_inE = new_recursive_definition
    false
    mt_env
    'IBAM_Opcode_inE'
    "IBAM_Opcode_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = IBAM_Opcode_in";;

let Rst_Opcode_inE = new_recursive_definition
    false
    mt_env
    'Rst_Opcode_inE'
    "Rst_Opcode_inE (MTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          MB_Opcode_in MB_Data_in IBAM_Opcode_in Rst_Opcode_in)
     = Rst_Opcode_in";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm mt_env);;

let PTEnv_Selectors_Work = prove_thm
   ('PTEnv_Selectors_Work',
    "!e:mt_env.
     e = (MTEnv (IB_Opcode_inE e) (IB_Addr_inE e) (IB_Data_inE e) (IB_BS_inE e)
                (IB_BE_inE e) (MB_Opcode_inE e) (MB_Data_inE e)
                (IBAM_Opcode_inE e) (Rst_Opcode_inE e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:mt_env" Env_CASES)
    THEN REWRITE_TAC [IB_Opcode_inE; IB_Addr_inE; IB_Data_inE; IB_BS_inE;
                      IB_BE_inE; MB_Opcode_inE; MB_Data_inE; IBAM_Opcode_inE;
                      Rst_Opcode_inE]
   );;
```

```
%-----------------------------------------------------------------------------
   Abstract data type for the output.
-----------------------------------------------------------------------------%

let mt_out =
   define_type 'mt_out'
               'mt_out = MTOut mbmop wordnn wordnn wordn ibsop wordnn';;

let MB_Opcode_out0 = new_recursive_definition
    false
    mt_out
    'MB_Opcode_out0'
    "MB_Opcode_out0 (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out
                           IB_Opcode_out IB_Data_out)
     = MB_Opcode_out";;

let MB_Addr_out0 = new_recursive_definition
    false
    mt_out
    'MB_Addr_out0'
    "MB_Addr_out0 (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out
                           IB_Opcode_out IB_Data_out)
     = MB_Addr_out";;

let MB_Data_out0 = new_recursive_definition
    false
    mt_out
    'MB_Data_out0'
    "MB_Data_out0 (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out
                           IB_Opcode_out IB_Data_out)
     = MB_Data_out";;

let MB_BS_out0 = new_recursive_definition
    false
    mt_out
    'MB_BS_out0'
    "MB_BS_out0 (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out
                           IB_Opcode_out IB_Data_out)
     = MB_BS_out";;

let IB_Opcode_out0 = new_recursive_definition
    false
    mt_out
    'IB_Opcode_out0'
    "IB_Opcode_out0 (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out
                           IB_Opcode_out IB_Data_out)
     = IB_Opcode_out";;

let IB_Data_out0 = new_recursive_definition
    false
    mt_out
    'IB_Data_out0'
    "IB_Data_out0 (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out
                           IB_Opcode_out IB_Data_out)
     = IB_Data_out";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm mt_out);;

let PTOut_Selectors_Work = prove_thm
   ('PTOut_Selectors_Work',
    "!p:mt_out.
     p = (MTOut (MB_Opcode_out0 p) (MB_Addr_out0 p) (MB_Data_out0 p)
                (MB_BS_out0 p) (IB_Opcode_out0 p) (IB_Data_out0 p))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "p:mt_out" Out_CASES)
    THEN REWRITE_TAC [MB_Opcode_out0; MB_Addr_out0; MB_Data_out0; MB_BS_out0;
                      IB_Opcode_out0; IB_Data_out0]
   );;

close_theory();;
```

```
%-----------------------------------------------------------------------------

   File:      mtransp_def.ml

   Author:    (c) D.A. Fura 1993

   Date:      1 March 1993

   This file contains the ml source for the trans-level specification of the
   M-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/mport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing',true);;

system 'rm mtransp_def.th';;

new_theory 'mtransp_def';;

loadf 'abs_theory';;

map new_parent ['mtauxp_def';'array_def';'wordn_def'];;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%-----------------------------------------------------------------------------
   Output definition for M-Port instructions.
-----------------------------------------------------------------------------%

let MT_Write_OF = new_definition
   ('MT_Write_OF',
    "! (rep :^REP_ty) (e :mt_env) .
     MT_Write_OF rep  e =
        let MB_Opcode_out = MBM_WriteLM in

        let bs = VAL 1 (IB_BS_inE e) in
        let a0 = IB_Addr_inE e in
        let a0_0 = ALTER ARBN (0) a0 in
        let a1_0 = ALTER a0_0 (1) (bs > 0 => (INCN 18 a0) | ARBN) in
        let a2_0 = ALTER a1_0 (2) (bs > 1 => (INCN 18 (INCN 18 a0)) | ARBN) in
        let a3_0 = ALTER a2_0 (3) (bs > 2 => (INCN 18 (INCN 18 (INCN 18 a0)))
                                          | ARBN) in
        let MB_Addr_out = a3_0 in

        let d0 = ELEMENT (IB_Data_inE e) (0) in
        let d1 = ELEMENT (IB_Data_inE e) (1) in
        let d2 = ELEMENT (IB_Data_inE e) (2) in
        let d3 = ELEMENT (IB_Data_inE e) (3) in
        let m0 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (0)) in
        let m1 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (1)) in
        let m2 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (2)) in
        let m3 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (3)) in
        let be0 = ELEMENT (IB_BE_inE e) (0) in
        let be1 = ELEMENT (IB_BE_inE e) (1) in
        let be2 = ELEMENT (IB_BE_inE e) (2) in
        let be3 = ELEMENT (IB_BE_inE e) (3) in

        let o00 = ELEMENT be0 (0) => SUBARRAY d0 (7,0) | SUBARRAY m0 (7,0) in
        let o01 = ELEMENT be0 (1) => SUBARRAY d0 (15,8) | SUBARRAY m0 (15,8) in
        let o02 = ELEMENT be0 (2) => SUBARRAY d0 (23,16) | SUBARRAY m0 (23,16) in
        let o03 = ELEMENT be0 (3) => SUBARRAY d0 (31,24) | SUBARRAY m0 (31,24) in
        let o10 = ELEMENT be1 (0) => SUBARRAY d1 (7,0) | SUBARRAY m1 (7,0) in
        let o11 = ELEMENT be1 (1) => SUBARRAY d1 (15,8) | SUBARRAY m1 (15,8) in
        let o12 = ELEMENT be1 (2) => SUBARRAY d1 (23,16) | SUBARRAY m1 (23,16) in
```

```
        let o13 = ELEMENT be1 (3) => SUBARRAY d1 (31,24) | SUBARRAY m1 (31,24) in
        let o20 = ELEMENT be2 (0) => SUBARRAY d2 (7,0) | SUBARRAY m2 (7,0) in
        let o21 = ELEMENT be2 (1) => SUBARRAY d2 (15,8) | SUBARRAY m2 (15,8) in
        let o22 = ELEMENT be2 (2) => SUBARRAY d2 (23,16) | SUBARRAY m2 (23,16) in
        let o23 = ELEMENT be2 (3) => SUBARRAY d2 (31,24) | SUBARRAY m2 (31,24) in
        let o30 = ELEMENT be3 (0) => SUBARRAY d3 (7,0) | SUBARRAY m3 (7,0) in
        let o31 = ELEMENT be3 (1) => SUBARRAY d3 (15,8) | SUBARRAY m3 (15,8) in
        let o32 = ELEMENT be3 (2) => SUBARRAY d3 (23,16) | SUBARRAY m3 (23,16) in
        let o33 = ELEMENT be3 (3) => SUBARRAY d3 (31,24) | SUBARRAY m3 (31,24) in

        let d00 = MALTER ARBN (7,0) o00 in
        let d01 = MALTER d00 (15,8) o01 in
        let d02 = MALTER d01 (23,16) o02 in
        let d03 = MALTER d02 (31,24) o03 in
        let d10 = MALTER ARBN (7,0) o10 in
        let d11 = MALTER d10 (15,8) o11 in
        let d12 = MALTER d11 (23,16) o12 in
        let d13 = MALTER d12 (31,24) o13 in
        let d20 = MALTER ARBN (7,0) o20 in
        let d21 = MALTER d20 (15,8) o21 in
        let d22 = MALTER d21 (23,16) o22 in
        let d23 = MALTER d22 (31,24) o23 in
        let d30 = MALTER ARBN (7,0) o30 in
        let d31 = MALTER d30 (15,8) o31 in
        let d32 = MALTER d31 (23,16) o32 in
        let d33 = MALTER d32 (31,24) o33 in

        let o0_0 = ALTER ARBN (0) (Ham_Enc rep d03) in
        let o1_0 = ALTER o0_0 (1) (bs > 0 => (Ham_Enc rep d13) | ARBN) in
        let o2_0 = ALTER o1_0 (2) (bs > 1 => (Ham_Enc rep d23) | ARBN) in
        let o3_0 = ALTER o2_0 (3) (bs > 2 => (Ham_Enc rep d33) | ARBN) in
        let MB_Data_out = o3_0 in
        let MB_BS_out = IB_BS_inE e in

        let IB_Opcode_out = IBS_Ready in
        let IB_Data_out = ARBN in

        (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out IB_Opcode_out
               IB_Data_out)"
    );;

let MT_Read_OF = new_definition
    ('MT_Read_OF',
     "! (rep :^REP_ty) (e :mt_env) .
      MT_Read_OF rep e =
        let MB_Opcode_out = MBM_ReadLM in
        let bs = VAL 1 (IB_BS_inE e) in
        let a0 = IB_Addr_inE e in
        let a0_0 = ALTER ARBN (0) a0 in
        let a1_0 = ALTER a0_0 (1) (bs > 0 => (INCN 18 a0) | ARBN) in
        let a2_0 = ALTER a1_0 (2) (bs > 1 => (INCN 18 (INCN 18 a0)) | ARBN) in
        let a3_0 = ALTER a2_0 (3) (bs > 2 => (INCN 18 (INCN 18 (INCN 18 a0)))
                                          | ARBN) in
        let MB_Addr_out = a3_0 in
        let MB_Data_out = ARBN in
        let MB_BS_out = IB_BS_inE e in
        let IB_Opcode_out = IBS_Ready in
        let d0 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (0)) in
        let d1 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (1)) in
        let d2 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (2)) in
        let d3 = Ham_Dec rep (ELEMENT (MB_Data_inE e) (3)) in
        let d0_0 = ALTER ARBN (0) d0 in
        let d1_0 = ALTER d0_0 (1) (bs > 0 => d1 | ARBN) in
        let d2_0 = ALTER d1_0 (2) (bs > 1 => d2 | ARBN) in
        let d3_0 = ALTER d2_0 (3) (bs > 2 => d3 | ARBN) in
        let IB_Data_out = d3_0 in

        (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out IB_Opcode_out
               IB_Data_out)"
    );;

let MT_Idle_OF = new_definition
```

```
   ('MT_Idle_OF',
    "! (rep :^REP_ty) (e :mt_env) .
     MT_Idle_OF rep e =
         let MB_Opcode_out = MBM_Idle in
         let MB_Addr_out = ARBN in
         let MB_Data_out = ARBN in
         let MB_BS_out = IB_BS_inE e in
         let IB_Opcode_out = IBS_Idle in
         let IB_Data_out = ARBN in

         (MTOut MB_Opcode_out MB_Addr_out MB_Data_out MB_BS_out IB_Opcode_out
                IB_Data_out)"
   );;

%-----------------------------------------------------------------------------
   M-Port interpreter definition.
-----------------------------------------------------------------------------%

let MT_Exec = new_definition
   ('MT_Exec',
    "! (mti :MTI) (e :timeT->mt_env) (p :timeT->mt_out) (t :timeT) .
     MT_Exec mti e p t =
         (MB_Opcode_inE (e t) = MBS_Ready) /\
         (IBAM_Opcode_inE (e t) = IBAM_ProcP) /\
         ((mti = MT_Write) => (IB_Opcode_inE (e t) = PBM_WriteLM) |
          (mti = MT_Read) =>  (IB_Opcode_inE (e t) = PBM_ReadLM)
        % (mti = MT_Idle) % | ((IB_Opcode_inE (e t) = PBM_WritePIU) \/
                               (IB_Opcode_inE (e t) = PBM_ReadPIU) \/
                               (IB_Opcode_inE (e t) = PBM_WriteCB) \/
                               (IB_Opcode_inE (e t) = PBM_ReadCB)))"
   );;

let MT_PreC = new_prim_rec_definition
   ('MT_PreC',
    "(MT_PreC (mti:MTI) e p 0 = T) /\
     (MT_PreC mti e p (SUC t) =
         (MT_Exec MT_Write e p t /\ MT_PreC MT_Write e p t) \/
         (MT_Exec MT_Read e p t /\ MT_PreC MT_Read e p t) \/
         (MT_Exec MT_Idle e p t /\ MT_PreC MT_Idle e p t))"
   );;

let MT_PostC = new_definition
   ('MT_PostC',
    "! (rep :^REP_ty) (mti :MTI) (e :timeT->mt_env)
       (p :timeT->mt_out) (t :timeT) .
     MT_PostC rep mti e p t =
         (mti = MT_Write) => (p t = MT_Write_OF rep (e t)) |
         (mti = MT_Read) => (p t = MT_Read_OF rep (e t))
       % (mti = MT_Idle) % | (p t = MT_Idle_OF rep (e t))"
   );;

let MT_Correct = new_definition
   ('MT_Correct',
    "! (rep :^REP_ty) (mti :MTI) (e :timeT->mt_env)
       (p :timeT->mt_out) (t :timeT) .
     MT_Correct rep mti e p t =
         MT_Exec mti e p t /\
         MT_PreC mti e p t
         ==>
         MT_PostC rep mti e p t"
   );;

let MTSet_Correct = new_definition
   ('MTSet_Correct',
    "! (rep :^REP_ty) (e :timeT->mt_env) (p :timeT->mt_out) .
     MTSet_Correct rep e p = !(mti:MTI)(t:timeT). MT_Correct rep mti e p t"
   );;

close_theory();;


%-----------------------------------------------------------------------------
```

```
   File:      mtabs_def.ml

   Author:    (c) D.A. Fura 1993

   Date:      27 January 1993

------------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/mport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/pport/pproc/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/tools/';
                                  '/home/elvis6/dfura/hol/ml/'
                                 ]);;

set_flag ('timing',true);;

system 'rm mtabs_def.th';;

new_theory 'mtabs_def';;

map new_parent ['piuaux_def';'mtaux_def';'maux_def';'array_def';'wordn_def';
                'busn_def';'templogic_def';'mtrans_def';'mclock_def'];;

new_type_abbrev ('time',":num");;
new_type_abbrev ('timeT',":num");;
new_type_abbrev ('timeC',":num");;

let ale_sig_ib = new_definition
   ('ale_sig_ib',
    "! (e' :timeC->mc_env) .
     ale_sig_ib e' =
        \u':timeC. BSel(I_hlda_E (e' u')) /\ (~BSel(I_male_E (e' u')) \/
                                              ~BSel(I_rale_E (e' u')) \/
                                              ~BSel(I_cale_E (e' u')))"
   );;

let rdy_sig_ib = new_definition
   ('rdy_sig_ib',
    "! (e' :timeC->mc_env) (p' :timeC->mc_out) .
     rdy_sig_ib e' p' =
        \u':timeC. (BSel(I_srdy_O (p' u')) = LO) /\ BSel(I_last_E (e' u'))"
   );;

let ack_sig_ib = new_definition
   ('ack_sig_ib',
    "! (e' :timeC->mc_env) (p' :timeC->mc_out) .
     ack_sig_ib e' p' =
        \u':timeC. (BSel(I_srdy_O (p' u')) = LO) /\ ~BSel(I_last_E (e' u'))"
   );;

let IB_Slave = new_definition
   ('IB_Slave',
    "IB_Slave (mti :MTI)
              (e :timeT->mt_env) (p :timeT->mt_out) (t :timeT)
              (e' :timeC->mc_env) (p' :timeC->mc_out) (ti' :timeC) =

       let write = (ELEMENT (BSel(I_ad_inE (e' ti'))) (27)) in
       let read = (~write) in
       let lmem = (BSel(I_male_E (e' ti')) = F) in
       let piu = (BSel(I_rale_E (e' ti')) = F) in
       let cbus = (BSel(I_cale_E (e' ti')) = F) in
       let valid_rqt = % p-port control signals valid %
           (lmem ==> (~piu /\ ~cbus) /\
            piu ==> ~cbus /\
           (?u':timeC.
             STABLE_TRUE_THEN_FALSE (bsig I_last_E e') (ti',u'))) in
       let valid_ack = % m-port control signals valid %
           (?u':timeC. STABLE_HI_THEN_LO (bsig I_srdy_O p') (ti',u') /\
            !u':timeC.
             rdy_sig_ib e' p' u' ==>
```

```
      (?v':timeC. STABLE_HI_THEN_LO (bsig I_srdy_O p') (u'+1,v'))) in
% end-of-active-transaction time %
let t'ack = @u':timeC. STABLE_FALSE_THEN_TRUE (ack_sig_ib e' p') (ti',u')
in
% slave-ready 0,1,2,3 times %
let t'rdy0 = @u':timeC. NTH_TIME_LO 0 (bsig I_srdy_O p') ti' u' in
let t'rdy1 = @u':timeC. NTH_TIME_LO 1 (bsig I_srdy_O p') ti' u' in
let t'rdy2 = @u':timeC. NTH_TIME_LO 2 (bsig I_srdy_O p') ti' u' in
let t'rdy3 = @u':timeC. NTH_TIME_LO 3 (bsig I_srdy_O p') ti' u' in
% data-valid predicates for write-data words 0,1,2,3 %
let dv0 = (t'rdy0 <= t'ack /\
           STABLE_AB (sig I_ad_inE e') (ti'+1,t'rdy0)) in
let dv1 = (t'rdy1 <= t'ack /\
           STABLE_AB (sig I_ad_inE e') (t'rdy0+1,t'rdy1)) in
let dv2 = (t'rdy2 <= t'ack /\
           STABLE_AB (sig I_ad_inE e') (t'rdy1+1,t'rdy2)) in
let dv3 = (t'rdy3 <= t'ack /\
           STABLE_AB (sig I_ad_inE e') (t'rdy2+1,t'rdy3)) in
% write-data words 0,1,2,3 %
let d0 = (dv0 => BSel(I_ad_inE (e' t'rdy0)) | ARBN) in
let d1 = (dv1 => BSel(I_ad_inE (e' t'rdy1)) | ARBN) in
let d2 = (dv2 => BSel(I_ad_inE (e' t'rdy2)) | ARBN) in
let d3 = (dv3 => BSel(I_ad_inE (e' t'rdy3)) | ARBN) in
% byte-enable-valid predicates for byte enables 1,2,3 %
let bv1 = (t'rdy0 <= t'ack /\
           STABLE_AB (sig I_be_E e') (ti'+1,t'rdy0)) in
let bv2 = (t'rdy1 <= t'ack /\
           STABLE_AB (sig I_be_E e') (t'rdy0+1,t'rdy1)) in
let bv3 = (t'rdy2 <= t'ack /\
           STABLE_AB (sig I_be_E e') (t'rdy1+1,t'rdy2)) in
% byte enables 0,1,2,3 %
let b0 = NOTN 3 (BSel(I_be_E (e' ti'))) in
let b1 = (bv1 => NOTN 3 (BSel(I_be_E (e' t'rdy0))) | ARBN) in
let b2 = (bv2 => NOTN 3 (BSel(I_be_E (e' t'rdy1))) | ARBN) in
let b3 = (bv3 => NOTN 3 (BSel(I_be_E (e' t'rdy2))) | ARBN) in
% data-valid predicates for read-data words 0,1,2,3 %
let ev0 = (t'rdy0 <= t'ack) in
let ev1 = (t'rdy1 <= t'ack) in
let ev2 = (t'rdy2 <= t'ack) in
let ev3 = (t'rdy3 <= t'ack) in
% read-data words 0,1,2,3 %
let e0 = (ev0 => wordnVAL (BSel(I_ad_outO (p' t'rdy0))) | ARBN) in
let e1 = (ev1 => wordnVAL (BSel(I_ad_outO (p' t'rdy1))) | ARBN) in
let e2 = (ev2 => wordnVAL (BSel(I_ad_outO (p' t'rdy2))) | ARBN) in
let e3 = (ev3 => wordnVAL (BSel(I_ad_outO (p' t'rdy3))) | ARBN) in

((IB_Opcode_outO (p t) =
    (lmem /\
     valid_ack /\
     (valid_rqt ==>
      (?t'ack:timeC. STABLE_TRUE_THEN_FALSE
                       (ack_sig_ib e' p') (ti',t'ack))) /\
     STABLE_AB_OFF (sig I_srdy_O p') (ti',ti') /\
     STABLE_AB_OFFn (sig I_ad_outO p') (ti',ti') /\
     (write ==>
        (!u':timeC.
         ti' < u' ==>
           STABLE_FALSE (ale_sig_ib e') (ti'+1,u') ==>
              STABLE_AB_OFFn (sig I_ad_outO p') (ti'+1,u'))) /\
     (!u':timeC.
       t'ack < u' ==>
         STABLE_FALSE (ale_sig_ib e') (t'ack+1,u') ==>
           (STABLE_AB_OFF (sig I_srdy_O p') (t'ack+1,u') /\
            STABLE_AB_OFFn (sig I_ad_outO p') (t'ack+1,u'))))
  => IBS_Ready |
    (~lmem /\
     STABLE_AB_OFF (sig I_srdy_O p') (ti',ti') /\
     STABLE_AB_OFFn (sig I_ad_outO p') (ti',ti') /\
     (!u':timeC.
      ti' < u' ==>
        STABLE_FALSE (ale_sig_ib e') (ti'+1,u') ==>
          (STABLE_AB_OFF (sig I_srdy_O p') (ti'+1,u') /\
```

```
                STABLE_AB_OFFn (sig I_ad_outO p') (ti'+1,u'))))
         => IBS_Idle | IBS_Illegal) /\
        (IB_Data_outO (p t) =
             ALTER (ALTER (ALTER (ALTER ARBN (0) e0) (1) e1) (2) e2) (3) e3) /\

        (IB_Opcode_inE (e t) =
           ((valid_ack ==> valid_rqt) /\ lmem /\ write) => PBM_WriteLM |
           ((valid_ack ==> valid_rqt) /\ lmem /\ read) => PBM_ReadLM |
           (piu /\ write) => PBM_WritePIU |
           (piu /\ read) => PBM_ReadPIU |
           (cbus /\ write) => PBM_WriteCB |
           (cbus /\ read) => PBM_ReadCB | PBM_Illegal) /\
        (IB_Addr_inE (e t) = SUBARRAY (BSel(I_ad_inE (e' ti'))) (23,0)) /\
        (IB_Data_inE (e t) =
             ALTER (ALTER (ALTER (ALTER ARBN (0) d0) (1) d1) (2) d2) (3) d3) /\
        (IB_BS_inE (e t) = SUBARRAY (BSel(I_ad_inE (e' ti'))) (25,24)) /\
        (IB_BE_inE (e t) =
             ALTER (ALTER (ALTER (ALTER ARBN (0) b0) (1) b1) (2) b2) (3) b3))"
   );;

let IBA_PMSlave = new_definition
   ('IBA_PMSlave',
    "IBA_PMSlave (mti :MTI)
                 (e :timeT->mt_env) (p :timeT->mt_out) (t :timeT)
                 (e' :timeC->mc_env) (p' :timeC->mc_out) (ti' :timeC) =
       let t'ack = @u':timeC. STABLE_FALSE_THEN_TRUE (ack_sig_ib e' p') (ti',u')
       in
       (IBAPM_Opcode_inE (e t) =
          (STABLE_AB_TRUE (sig I_hlda_E e') (ti',t'ack))
           => IBAPM_ProcP | IBAPM_Illegal)"
   );;

let cs_sig_mb = new_definition
   ('cs_sig_mb',
    "! (p' :timeC->mc_out) .
     cs_sig_mb p' =
        \u':timeC. ~ASel(MB_cs_eeprom_O (p' u')) \/ ~ASel(MB_cs_sram_O (p' u'))"
   );;

let MB_Master = new_definition
   ('MB_Master',
    "MB_Master (mti :MTI)
               (e :timeT->mt_env) (p :timeT->mt_out) (t :timeT)
               (e' :timeC->mc_env) (p' :timeC->mc_out) (ti' :timeC) =
     % data-write times %
     let t'w0 = @u':timeC. NTH_TIME_CHANGES_FALSE 0 (bsig MB_we_O p') ti' u' in
     let t'w1 = @u':timeC. NTH_TIME_CHANGES_FALSE 1 (bsig MB_we_O p') ti' u' in
     let t'w2 = @u':timeC. NTH_TIME_CHANGES_FALSE 2 (bsig MB_we_O p') ti' u' in
     let t'w3 = @u':timeC. NTH_TIME_CHANGES_FALSE 3 (bsig MB_we_O p') ti' u' in
     % data-read times %
     let t'r0 = @u':timeC. NTH_TIME_CHANGES_FALSE 0 (bsig MB_oe_O p') ti' u' in
     let t'r1 = @u':timeC. NTH_TIME_CHANGES_FALSE 1 (bsig MB_oe_O p') ti' u' in
     let t'r2 = @u':timeC. NTH_TIME_CHANGES_FALSE 2 (bsig MB_oe_O p') ti' u' in
     let t'r3 = @u':timeC. NTH_TIME_CHANGES_FALSE 3 (bsig MB_oe_O p') ti' u' in
     let sram = ~BSel(MB_cs_sram_O (p' (ti'+1))) in
     let eeprom= ~BSel(MB_cs_eeprom_O (p' (ti'+1))) in
     % address-valid predicate for all address words %
     let valid_addr =
        ((sram ==>
            (!u'. (CHANGES_FALSE (bsig MB_we_O p') u' \/
                   CHANGES_FALSE (bsig MB_oe_O p') u')
                    ==> STABLE_AB (sig MB_addrO p') (u',u'+1))) /\
         (eeprom ==>
            (!u'. (CHANGES_FALSE (bsig MB_we_O p') u' \/
                   CHANGES_FALSE (bsig MB_oe_O p') u')
                    ==> STABLE_AB (sig MB_addrO p') (u',u'+2))) /\
         (!u' v' n. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                     STABLE_FALSE (ale_sig_ib e') (ti'+1,v') /\
                     NTH_TIME_CHANGES_FALSE n (bsig MB_oe_O p') ti' u' /\
                     NTH_TIME_CHANGES_FALSE n (bsig MB_we_O p') ti' v')
                      ==> (BSel(MB_addrO (p' u')) = BSel(MB_addrO (p' v'))))))) in
     % data-valid predicate for all data words %
```

```
let valid_data =
   ((sram ==>
       ((!u'. (CHANGES_FALSE (bsig MB_we_O p') u'
               ==> STABLE_AB (sig MB_data_outO p') (u',u'+1)))) /\
        (!u'. (~ASel(MB_oe_O (p' u')) \/ ~BSel(MB_oe_O (p' u')))
               ==> STABLE_AB_OFFn (sig MB_data_outO p') (u',u'))) /\
    (eeprom ==>
       ((!u'. (CHANGES_FALSE (bsig MB_we_O p') u'
               ==> STABLE_AB (sig MB_data_outO p') (u',u'+2)))) /\
        (!u'. (~ASel(MB_oe_O (p' u')) \/ ~BSel(MB_oe_O (p' u')))
               ==> STABLE_AB_OFFn (sig MB_data_outO p') (u',u*)))) in
% address words 0,1,2,3 %
let a0 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 0 (bsig MB_we_O p') ti' u')))
                 => BSel(MB_addrO (p' t'w0)) |
          (?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 0 (bsig MB_oe_O p') ti' u')))
                 => BSel(MB_addrO (p' t'r0)) | ARBN) in
let a1 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 1 (bsig MB_we_O p') ti' u')))
                 => BSel(MB_addrO (p' t'w1)) |
          (?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 1 (bsig MB_oe_O p') ti' u')))
                 => BSel(MB_addrO (p' t'r1)) | ARBN) in
let a2 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 2 (bsig MB_we_O p') ti' u')))
                 => BSel(MB_addrO (p' t'w2)) |
          (?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 2 (bsig MB_oe_O p') ti' u')))
                 => BSel(MB_addrO (p' t'r2)) | ARBN) in
let a3 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 3 (bsig MB_we_O p') ti' u')))
                 => BSel(MB_addrO (p' t'w3)) |
          (?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                (NTH_TIME_CHANGES_FALSE 3 (bsig MB_oe_O p') ti' u')))
                 => BSel(MB_addrO (p' t'r3)) | ARBN) in
% write-data words 0,1,2,3 %
let dw0 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 0 (bsig MB_we_O p') ti' u')))
                  => wordnVAL (BSel(MB_data_outO (p' t'w0))) | ARBN) in
let dw1 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 1 (bsig MB_we_O p') ti' u')))
                  => wordnVAL (BSel(MB_data_outO (p' t'w1))) | ARBN) in
let dw2 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 2 (bsig MB_we_O p') ti' u')))
                  => wordnVAL (BSel(MB_data_outO (p' t'w2))) | ARBN) in
let dw3 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 3 (bsig MB_we_O p') ti' u')))
                  => wordnVAL (BSel(MB_data_outO (p' t'w3))) | ARBN) in
% read-data words 0,1,2,3 %
let dr0 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 0 (bsig MB_oe_O p') ti' u')))
                  => (sram => BSel(MB_data_inE (e' (t'r0+1)))
                              | BSel(MB_data_inE (e' (t'r0+2)))) | ARBN) in
let dr1 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 1 (bsig MB_oe_O p') ti' u')))
                  => (sram => BSel(MB_data_inE (e' (t'r1+1)))
                              | BSel(MB_data_inE (e' (t'r1+2)))) | ARBN) in
let dr2 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 2 (bsig MB_oe_O p') ti' u')))
                  => (sram => BSel(MB_data_inE (e' (t'r2+1)))
                              | BSel(MB_data_inE (e' (t'r2+2)))) | ARBN) in
let dr3 = ((?u'. (STABLE_FALSE (ale_sig_ib e') (ti'+1,u') /\
                 (NTH_TIME_CHANGES_FALSE 3 (bsig MB_oe_O p') ti' u')))
                  => (sram => BSel(MB_data_inE (e' (t'r3+1)))
                              | BSel(MB_data_inE (e' (t'r3+2)))) | ARBN) in

((MB_Opcode_outO (p t) =
    (((valid_addr) /\
      (valid_data) /\
      (?u'. ~BSel(MB_we_O (p' u')) /\
            STABLE_FALSE (ale_sig_ib e') (ti'+1,u'))) => MBM_WriteLM |
```

```
          ((valid_addr) /\
           (valid_data) /\
           (?u'. ~BSel(MB_oe_O (p' u')) /\
                 STABLE_FALSE (ale_sig_ib e') (ti'+1,u'))) => MBM_ReadLM |
          (!u'. STABLE_FALSE (ale_sig_ib e') (ti'+1,u')
                ==> (STABLE_AB_TRUE (sig MB_cs_eeprom_O p') (ti'+1,u') /\
                     STABLE_AB_TRUE (sig MB_cs_sram_O p') (ti'+1,u') /\
                     STABLE_AB_TRUE (sig MB_we_O p') (ti'+1,u') /\
                     STABLE_AB_TRUE (sig MB_oe_O p') (ti'+1,u'))) => MBM_Idle
          | MBM_Illegal)) /\
      (MB_Addr_outO (p t) =
          ALTER (ALTER (ALTER (ALTER ARBN (0) a0) (1) a1) (2) a2) (3) a3) /\
      (MB_Data_outO (p t) =
          ALTER (ALTER (ALTER (ALTER ARBN (0) dw0) (1) dw1) (2) dw2) (3) dw3) /\
      (MB_BS_outO (p t) =
         (!u'. STABLE_FALSE (ale_sig_ib e') (ti'+1,u')
               ==> (LESS_THAN_N_TIMES_CHANGES_FALSE 1 (bsig MB_we_O p') ti' u' /\
                    LESS_THAN_N_TIMES_CHANGES_FALSE 1 (bsig MB_oe_O p') ti' u'))
          => WORDN 0 |
         (!u'. STABLE_FALSE (ale_sig_ib e') (ti'+1,u')
               ==> (LESS_THAN_N_TIMES_CHANGES_FALSE 2 (bsig MB_we_O p') ti' u' /\
                    LESS_THAN_N_TIMES_CHANGES_FALSE 2 (bsig MB_oe_O p') ti' u'))
          => WORDN 1 |
         (!u'. STABLE_FALSE (ale_sig_ib e') (ti'+1,u')
               ==> (LESS_THAN_N_TIMES_CHANGES_FALSE 3 (bsig MB_we_O p') ti' u' /\
                    LESS_THAN_N_TIMES_CHANGES_FALSE 3 (bsig MB_oe_O p') ti' u'))
          => WORDN 2 |
         (!u'. STABLE_FALSE (ale_sig_ib e') (ti'+1,u')
               ==> (LESS_THAN_N_TIMES_CHANGES_FALSE 4 (bsig MB_we_O p') ti' u' /\
                    LESS_THAN_N_TIMES_CHANGES_FALSE 4 (bsig MB_oe_O p') ti' u'))
          => WORDN 3 | ARB) /\

      (MB_Opcode_inE (e t) = MBS_Ready) /\
      (MB_Data_inE (e t) =
          ALTER (ALTER (ALTER (ALTER ARBN (0) dr0) (1) dr1) (2) dr2) (3) dr3))"
   );;


let Rst_Slave = new_definition
   ('Rst_Slave',
    "Rst_Slave (mti :MTI) (e :timeT->mt_env) (t :timeT) (e' :timeC->mc_env) =
        Rst_Opcode_inE (e t) =
           (!u':timeC. BSel (RstE (e' u')) = F) => RM_NoReset | RM_Illegal"
   );;

let MStateAbs = new_definition
   ('MStateAbs',
    "MStateAbs (mti :MTI) (s :timeT->mt_state) (e :timeT->mt_env)
               (p :timeT->mt_out) (t :timeT) (s' :timeC->mc_state)
               (e' :timeC->mc_env) (p' :timeC->mc_out) (ti' :timeC) =

       % (t' = 0) %
        (M_fsm_stateS (s' 0) = MI) /\
        ((ti' > 0) ==>
           (M_fsm_stateS (s' ti') = MT_fsm_stateS (s t))) /\

        (!ti'suc:timeC.
         NTH_TIME_TRUE (SUC t) (ale_sig_ib e') 0 ti'suc ==>
           (MT_fsm_stateS (s (t+1)) = M_fsm_stateS (s' ti'suc)))"
   );;

let MTAbs = new_definition
   ('MTAbs',
    "MTAbs (mti :MTI) (s :timeT->mt_state) (e :timeT->mt_env)
           (p :timeT->mt_out) (t :timeT) (s' :timeC->mc_state)
           (e' :timeC->mc_env) (p' :timeC->mc_out) =

        MT_Exec mti s e p t
        ==>
        (? (ti' :timeC).
           NTH_TIME_TRUE t (ale_sig_ib e') 0 ti' /\
           IB_Slave mti e p t e' p' ti' /\
```

```
          IBA_PMSlave mti e p t e' p' ti' /\
          MB_Master mti e p t e' p' ti' /\
          Rst_Slave mti e t e' /\
          MStateAbs mti s e p t s' e' p' ti')"
   );;

let MTAbsSet = new_definition
   ('MTAbsSet',
    "MTAbsSet (s :timeT->mt_state) (e :timeT->mt_env) (p :timeT->mt_out)
              (s' :timeC->mc_state) (e' :timeC->mc_env) (p' :timeC->mc_out) =
        !(mti:MTI)(t:timeT). MTAbs mti s e p t s' e' p'"
   );;

close_theory();;
```

4.4 C-Port Transaction-Level Specification

This section contains the theories *ctauxp_def* and *ctransp_def*, defining the C-Port transaction-level data structures and interpreter, respectively.

```
%-----------------------------------------------------------------------------

   File:        ctauxp_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        1 March 1993

   This file contains types and definitions for the transaction-level
   specification of the P-Process of the PIU C-Port.

-----------------------------------------------------------------------------%

set_flag ('timing', true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/ftep/piu/hol/cport/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm ctauxp_def.th';;

new_theory 'ctauxp_def';;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('timeT', ":num");;
new_type_abbrev ('wordn', ":(num->bool)");;
new_type_abbrev ('wordnn', ":(num->wordn)");;

%-----------------------------------------------------------------------------
   Abstract data type for the C-Port instruction set.
-----------------------------------------------------------------------------%

let CTI =
   define_type 'CTI'
               'CTI = CT_Write | CT_Read | CT_Idle';;

%-----------------------------------------------------------------------------
   Abstract data type for the C-Port transaction opcodes.
-----------------------------------------------------------------------------%

% P-Bus Master Opcodes %
let pbmop =
   define_type 'pbmop'
               'pbmop = PBM_WriteLM | PBM_WritePIU | PBM_WriteCB | PBM_ReadLM |
                        PBM_ReadPIU | PBM_ReadCB | PBM_Illegal';;
```

```
% I-Bus Slave Opcodes %
let ibsop =
   define_type 'ibsop'
               'ibsop = IBS_Ready | IBS_Idle | IBS_Illegal';;

% I-Bus Arbitration-Master Opcodes %
let ibamop =
   define_type 'ibamop'
               'ibamop = IBAM_ProcP | IBAM_ProcC | IBAM_Illegal';;

% C-Bus Master Opcodes %
let cbmop =
   define_type 'cbmop'
               'cbmop = CBM_WriteCB | CBM_ReadCB | CBM_Idle | CBM_Illegal';;

% C-Bus Slave Opcodes %
let cbsop =
   define_type 'cbsop'
               'cbsop = CBS_Ready | CBS_Illegal';;

% Reset Master Opcodes %
let rmop =
   define_type 'rmop'
               'rmop = RM_NoReset | RM_Illegal';;

%-----------------------------------------------------------------------------
   Abstract data type for the environment.
-----------------------------------------------------------------------------%

let ct_env =
   define_type 'ct_env'
               'ct_env = CTEnv pbmop wordn wordnn wordn wordnn cbsop wordnn';;

let IB_Opcode_inE = new_recursive_definition
    false
    ct_env
    'IB_Opcode_inE'
    "IB_Opcode_inE (CTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in
                          IB_BE_in CB_Opcode_in CB_Data_in)
     = IB_Opcode_in";;

let IB_Addr_inE = new_recursive_definition
    false
    ct_env
    'IB_Addr_inE'
    "IB_Addr_inE (CTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in
                          IB_BE_in CB_Opcode_in CB_Data_in)
     = IB_Addr_in";;

let IB_Data_inE = new_recursive_definition
    false
    ct_env
    'IB_Data_inE'
    "IB_Data_inE (CTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in
                          IB_BE_in CB_Opcode_in CB_Data_in)
     = IB_Data_in";;

let IB_BS_inE = new_recursive_definition
    false
    ct_env
    'IB_BS_inE'
    "IB_BS_inE (CTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in
                          IB_BE_in CB_Opcode_in CB_Data_in)
     = IB_BS_in";;

let IB_BE_inE = new_recursive_definition
    false
    ct_env
    'IB_BE_inE'
    "IB_BE_inE (CTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in
                          IB_BE_in CB_Opcode_in CB_Data_in)
```

```
     = IB_BE_in";;

let CB_Opcode_inE = new_recursive_definition
    false
    ct_env
    'CB_Opcode_inE'
    "CB_Opcode_inE (CTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in
                          IB_BE_in CB_Opcode_in CB_Data_in)
     = CB_Opcode_in";;

let CB_Data_inE = new_recursive_definition
    false
    ct_env
    'CB_Data_inE'
    "CB_Data_inE (CTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in
                          IB_BE_in CB_Opcode_in CB_Data_in)
     = CB_Data_in";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm ct_env);;

let CTEnv_Selectors_Work = prove_thm
   ('CTEnv_Selectors_Work',
    "!e:ct_env .
      e = (CTEnv (IB_Opcode_inE e) (IB_Addr_inE e) (IB_Data_inE e) (IB_BS_inE e)
                 (IB_BE_inE e) (CB_Opcode_inE e) (CB_Data_inE e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:ct_env" Env_CASES)
    THEN REWRITE_TAC [IB_Opcode_inE; IB_Addr_inE; IB_Data_inE; IB_BS_inE;
                      IB_BE_inE; CB_Opcode_inE; CB_Data_inE]
   );;

%----------------------------------------------------------------------------
   Abstract data type for the output.
----------------------------------------------------------------------------%

let ct_out =
   define_type 'ct_out'
               'ct_out = CTOut cbmop wordn wordnn wordn wordnn ibsop
                               wordnn';;

let CB_Opcode_outO = new_recursive_definition
    false
    ct_out
    'CB_Opcode_outO'
    "CB_Opcode_outO (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                           CB_BE_out IB_Opcode_out IB_Data_out)
     = CB_Opcode_out";;

let CB_Addr_outO = new_recursive_definition
    false
    ct_out
    'CB_Addr_outO'
    "CB_Addr_outO (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                           CB_BE_out IB_Opcode_out IB_Data_out)
     = CB_Addr_out";;

let CB_Data_outO = new_recursive_definition
    false
    ct_out
    'CB_Data_outO'
    "CB_Data_outO (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                           CB_BE_out IB_Opcode_out IB_Data_out)
     = CB_Data_out";;

let CB_BS_outO = new_recursive_definition
    false
    ct_out
    'CB_BS_outO'
    "CB_BS_outO (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                           CB_BE_out IB_Opcode_out IB_Data_out)
     = CB_BS_out";;
```

```
let CB_BE_out0 = new_recursive_definition
    false
    ct_out
    'CB_BE_out0'
    "CB_BE_out0 (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                       CB_BE_out IB_Opcode_out IB_Data_out)
     = CB_BE_out";;

let IB_Opcode_out0 = new_recursive_definition
    false
    ct_out
    'IB_Opcode_out0'
    "IB_Opcode_out0 (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                       CB_BE_out IB_Opcode_out IB_Data_out)
     = IB_Opcode_out";;

let IB_Data_out0 = new_recursive_definition
    false
    ct_out
    'IB_Data_out0'
    "IB_Data_out0 (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out
                       CB_BE_out IB_Opcode_out IB_Data_out)
     = IB_Data_out";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm ct_out);;

let CTOut_Selectors_Work = prove_thm
   ('CTOut_Selectors_Work',
    "!p:ct_out .
      p = (CTOut (CB_Opcode_out0 p) (CB_Addr_out0 p) (CB_Data_out0 p)
                 (CB_BS_out0 p) (CB_BE_out0 p) (IB_Opcode_out0 p)
                 (IB_Data_out0 p))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "p:ct_out" Out_CASES)
    THEN REWRITE_TAC [CB_Opcode_out0; CB_Addr_out0; CB_Data_out0; CB_BS_out0;
                      CB_BE_out0; IB_Opcode_out0; IB_Data_out0]
   );;

close_theory();;


%-----------------------------------------------------------------------------

   File:        ctransp_def.ml

   Author:      (c) D.A. Fura 1992-93

   Date:        1 March 1993

   This file contains the ml source for the trans-level specification of the
   C-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.  This specification is for the
   process associated with the P-Port - the C-Port masters the C-Bus according
   to a P-Port request.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/cport/pproc/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;
set_flag ('timing', true);;

system 'rm ctransp_def.th';;

new_theory 'ctransp_def';;

loadf 'abs_theory';;
```

```
map new_parent ['ctauxp_def';'array_def';'wordn_def';'ineq'];;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%----------------------------------------------------------------------------
   Output definition for C-Port instructions.
----------------------------------------------------------------------------%

let CT_Write_OF = new_definition
   ('CT_Write_OF',
    "! (rep :^REP_ty) (e :ct_env) .
     CT_Write_OF rep e =
        let CB_Opcode_out = CBM_WriteCB in
        let CB_Addr_out = IB_Addr_inE e in
        let bs = VAL 1 (IB_BS_inE e) in
        let d0 = ELEMENT (IB_Data_inE e) (0) in
        let d1 = ELEMENT (IB_Data_inE e) (1) in
        let d2 = ELEMENT (IB_Data_inE e) (2) in
        let d3 = ELEMENT (IB_Data_inE e) (3) in
        let o0 = ALTER ARBN (0) (Par_Enc rep d0) in
        let o1 = ALTER o0 (1) (bs > 0 => (Par_Enc rep d1) | ARBN) in
        let o2 = ALTER o1 (2) (bs > 1 => (Par_Enc rep d2) | ARBN) in
        let o3 = ALTER o2 (3) (bs > 2 => (Par_Enc rep d3) | ARBN) in
        let CB_Data_out = o3 in
        let CB_BS_out = IB_BS_inE e in
        let CB_BE_out = IB_BE_inE e in
        let IB_Opcode_out = IBS_Ready in
        let IB_Data_out = (ARBN:num->wordn) in

        (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out CB_BE_out
               IB_Opcode_out IB_Data_out)"
   );;

let CT_Read_OF = new_definition
   ('CT_Read_OF',
    "! (rep :^REP_ty) (e :ct_env) .
     CT_Read_OF rep e =
        let CB_Opcode_out = CBM_ReadCB in
        let CB_Addr_out = IB_Addr_inE e in
        let CB_Data_out = (ARBN:num->wordn) in
        let CB_BS_out = IB_BS_inE e in
        let CB_BE_out = IB_BE_inE e in
        let IB_Opcode_out = IBS_Ready in
        let bs = VAL 1 (IB_BS_inE e) in
        let d0 = Par_Dec rep (ELEMENT (CB_Data_inE e) (0)) in
        let d1 = Par_Dec rep (ELEMENT (CB_Data_inE e) (1)) in
        let d2 = Par_Dec rep (ELEMENT (CB_Data_inE e) (2)) in
        let d3 = Par_Dec rep (ELEMENT (CB_Data_inE e) (3)) in
        let d0_0 = ALTER ARBN (0) d0 in
        let d1_0 = ALTER d0_0 (1) (bs > 0 => d1 | ARBN) in
        let d2_0 = ALTER d1_0 (2) (bs > 1 => d2 | ARBN) in
        let d3_0 = ALTER d2_0 (3) (bs > 2 => d3 | ARBN) in
        let IB_Data_out = d3_0 in

        (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out CB_BE_out
               IB_Opcode_out IB_Data_out)"
   );;

let CT_Idle_OF = new_definition
   ('CT_Idle_OF',
    "! (rep :^REP_ty) (e :ct_env) .
     CT_Idle_OF rep e =
        let CB_Opcode_out = CBM_Idle in
        let CB_Addr_out = ARBN in
        let CB_Data_out = (ARBN:num->wordn) in
        let CB_BS_out = ARBN in
        let CB_BE_out = ARBN in
        let IB_Opcode_out = IBS_Ready in
        let IB_Data_out = ARBN in

        (CTOut CB_Opcode_out CB_Addr_out CB_Data_out CB_BS_out CB_BE_out
               IB_Opcode_out IB_Data_out)"
```

```
   );;

%-----------------------------------------------------------------------------
   C-Port interpreter definition.
-----------------------------------------------------------------------------%
let CT_Exec = new_definition
   ('CT_Exec',
    "! (cti :CTI) (e :timeT->ct_env) (p :timeT->ct_out) (t :timeT) .
     CT_Exec cti e p t =
        (CB_Opcode_inE (e t) = CBS_Ready) /\
        (IBAM_Opcode_inE (e t) = IBAM_ProcP) /\
        ((cti = CT_Write) => (IB_Opcode_inE (e t) = PBM_WriteCB) |
         (cti = CT_Read) =>  (IB_Opcode_inE (e t) = PBM_ReadCB)
       % (cti = CT_Idle) % | ((IB_Opcode_inE (e t) = PBM_WriteLM) \/
                              (IB_Opcode_inE (e t) = PBM_ReadLM) \/
                              (IB_Opcode_inE (e t) = PBM_WritePIU) \/
                              (IB_Opcode_inE (e t) = PBM_ReadPIU)))"
   );;

let CT_PreC = new_prim_rec_definition
   ('CT_PreC',
    "(CT_PreC (cti:CTI) e p 0 = T) /\
     (CT_PreC cti e p (SUC t) =
        (CT_Exec CT_Write e p t /\ CT_PreC CT_Write e p t) \/
        (CT_Exec CT_Read e p t /\ CT_PreC CT_Read e p t) \/
        (CT_Exec CT_Idle e p t /\ CT_PreC CT_Idle e p t))"
   );;

let CT_PostC = new_definition
   ('CT_PostC',
    "! (rep :^REP_ty) (cti :CTI) (e :timeT->ct_env)
       (p :timeT->ct_out) (t :timeT) .
     CT_PostC rep cti e p t =
        (cti = CT_Write) => (p t = CT_Write_OF rep (e t)) |
        (cti = CT_Read) => (p t = CT_Read_OF rep (e t))
      % (cti = CT_Idle) % | (p t = CT_Idle_OF rep (e t))"
   );;

let CT_Correct = new_definition
   ('CT_Correct',
    "! (rep :^REP_ty) (cti :CTI) (e :timeT->ct_env)
       (p :timeT->ct_out) (t :timeT) .
     CT_Correct rep cti e p t =
        CT_Exec cti e p t /\
        CT_PreC cti e p t
        ==>
        CT_PostC rep cti e p t"
   );;

let CTSet_Correct = new_definition
   ('CTSet_Correct',
    "! (rep :^REP_ty) (e :timeT->ct_env) (p :timeT->ct_out) .
     CTSet_Correct rep e p = !(cti:CTI)(t:timeT). CT_Correct rep cti e p t"
   );;

close_theory();;
```

4.5 R-Port Transaction-Level Specification

This section contains the theories *rtauxp_def* and *rtransp_def*, defining the R-Port transaction-level data structures and interpreter, respectively.

```
%-----------------------------------------------------------------------------

   File:      rtauxp_def.ml
```

```
   Author:   (c) D.A. Fura 1993

   Date:     1 March 1993

   This file contains types and definitions for the transaction-level
   specification of the P-Process of the PIU R-Port.

------------------------------------------------------------------------%

set_flag ('timing',true);;

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/ftep/piu/hol/rport/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

system 'rm rtauxp_def.th';;

new_theory 'rtauxp_def';;

map new_parent ['raux_def'];;

new_type_abbrev ('time', ":num");;
new_type_abbrev ('timeT', ":num");;
new_type_abbrev ('wordn', ":num->bool");;
new_type_abbrev ('wordnn', ":num->wordn");;

%-------------------------------------------------------------------------
   Abstract data type for the R-port instruction set.
------------------------------------------------------------------------%

let RTI =
   define_type 'RTI'
               'RTI = RT_Write | RT_Read | RT_Idle';;

%-------------------------------------------------------------------------
   Abstract data type for the R-port transaction opcodes.
------------------------------------------------------------------------%

% P-Bus Master Opcodes %
let pbmop =
   define_type 'pbmop'
               'pbmop = PBM_WriteLM | PBM_WritePIU | PBM_WriteCB | PBM_ReadLM |
                        PBM_ReadPIU | PBM_ReadCB | PBM_Illegal';;

% I-Bus Slave Opcodes %
let ibsop =
   define_type 'ibsop'
               'ibsop = IBS_Ready | IBS_Idle | IBS_Illegal';;

% I-Bus Arbitration-Master Opcodes %
let ibamop =
   define_type 'ibamop'
               'ibamop = IBAM_ProcP | IBAM_ProcC | IBAM_Illegal';;

% Reset Master Opcodes %
let rmop =
   define_type 'rmop'
               'rmop = RM_NoReset | RM_Illegal';;

%-------------------------------------------------------------------------
   Abstract data type for the state.
------------------------------------------------------------------------%

let rt_state =
   define_type 'rt_state'
               'rt_state = RTState wordn wordn wordn wordn wordn wordn wordn
                                  wordn wordn wordn wordn wordn';;

let RT_icrS = new_recursive_definition
    false
    rt_state
```

```
    'RT_icrS'
    "RT_icrS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in RT_ctr2_in
                     RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_icr";;

let RT_gcrS = new_recursive_definition
    false
    rt_state
    'RT_gcrS'
    "RT_gcrS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in RT_ctr2_in
                     RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_gcr";;

let RT_ccrS = new_recursive_definition
    false
    rt_state
    'RT_ccrS'
    "RT_ccrS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in RT_ctr2_in
                     RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ccr";;

let RT_srS = new_recursive_definition
    false
    rt_state
    'RT_srS'
    "RT_srS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in RT_ctr2_in
                     RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_sr";;

let RT_ctr0_inS = new_recursive_definition
    false
    rt_state
    'RT_ctr0_inS'
    "RT_ctr0_inS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                     RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr0_in";;

let RT_ctr1_inS = new_recursive_definition
    false
    rt_state
    'RT_ctr1_inS'
    "RT_ctr1_inS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                     RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr1_in";;

let RT_ctr2_inS = new_recursive_definition
    false
    rt_state
    'RT_ctr2_inS'
    "RT_ctr2_inS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                     RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr2_in";;

let RT_ctr3_inS = new_recursive_definition
    false
    rt_state
    'RT_ctr3_inS'
    "RT_ctr3_inS (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                     RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr3_in";;

let RT_ctr0S = new_recursive_definition
    false
    rt_state
    'RT_ctr0S'
    "RT_ctr0S (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                     RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr0";;

let RT_ctr1S = new_recursive_definition
    false
    rt_state
```

```
    'RT_ctr1S'
    "RT_ctr1S (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                      RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr1";;

let RT_ctr2S = new_recursive_definition
    false
    rt_state
    'RT_ctr2S'
    "RT_ctr2S (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                      RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr2";;

let RT_ctr3S = new_recursive_definition
    false
    rt_state
    'RT_ctr3S'
    "RT_ctr3S (RTState RT_icr RT_gcr RT_ccr RT_sr RT_ctr0_in RT_ctr1_in
                      RT_ctr2_in RT_ctr3_in RT_ctr0 RT_ctr1 RT_ctr2 RT_ctr3)
    = RT_ctr3";;

let State_CASES =
   prove_cases_thm (prove_induction_thm rt_state);;

let State_Selectors_Work = prove_thm
   ('State_Selectors_Work',
    "!s:rt_state.
      s = (RTState (RT_icrS s) (RT_gcrS s) (RT_ccrS s) (RT_srS s) (RT_ctr0_inS s)
                   (RT_ctr1_inS s) (RT_ctr2_inS s) (RT_ctr3_inS s) (RT_ctr0S s)
                   (RT_ctr1S s) (RT_ctr2S s) (RT_ctr3S s))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "s:rt_state" State_CASES)
    THEN REWRITE_TAC [RT_icrS;RT_gcrS;RT_ccrS;RT_srS;RT_ctr0_inS;RT_ctr1_inS;
                      RT_ctr2_inS;RT_ctr3_inS;RT_ctr0S;RT_ctr1S;RT_ctr2S;
                      RT_ctr3S]
   );;

%----------------------------------------------------------------------------
   Abstract data type for the environment.
----------------------------------------------------------------------------%

let rt_env =
   define_type 'rt_env'
               'rt_env = RTEnv pbmop wordn wordnn wordn wordnn
                               ibamop
                               rmop';;

let IB_Opcode_inE = new_recursive_definition
    false
    rt_env
    'IB_Opcode_inE'
    "IB_Opcode_inE (RTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          IBAM_Opcode_in Rst_Opcode_in)
     = IB_Opcode_in";;

let IB_Addr_inE = new_recursive_definition
    false
    rt_env
    'IB_Addr_inE'
    "IB_Addr_inE (RTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          IBAM_Opcode_in Rst_Opcode_in)
     = IB_Addr_in";;

let IB_Data_inE = new_recursive_definition
    false
    rt_env
    'IB_Data_inE'
    "IB_Data_inE (RTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          IBAM_Opcode_in Rst_Opcode_in)
     = IB_Data_in";;

let IB_BS_inE = new_recursive_definition
```

```
    false
    rt_env
    'IB_BS_inE'
    "IB_BS_inE (RTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          IBAM_Opcode_in Rst_Opcode_in)
     = IB_BS_in";;

let IB_BE_inE = new_recursive_definition
    false
    rt_env
    'IB_BE_inE'
    "IB_BE_inE (RTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          IBAM_Opcode_in Rst_Opcode_in)
     = IB_BE_in";;

let IBAM_Opcode_inE = new_recursive_definition
    false
    rt_env
    'IBAM_Opcode_inE'
    "IBAM_Opcode_inE (RTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          IBAM_Opcode_in Rst_Opcode_in)
     = IBAM_Opcode_in";;

let Rst_Opcode_inE = new_recursive_definition
    false
    rt_env
    'Rst_Opcode_inE'
    "Rst_Opcode_inE (RTEnv IB_Opcode_in IB_Addr_in IB_Data_in IB_BS_in IB_BE_in
                          IBAM_Opcode_in Rst_Opcode_in)
     = Rst_Opcode_in";;

let Env_CASES =
   prove_cases_thm (prove_induction_thm rt_env);;

let Env_Selectors_Work = prove_thm
   ('Env_Selectors_Work',
    "!e:rt_env.
     e = (RTEnv (IB_Opcode_inE e) (IB_Addr_inE e) (IB_Data_inE e) (IB_BS_inE e)
                (IB_BE_inE e) (IBAM_Opcode_inE e) (Rst_Opcode_inE e))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "e:rt_env" Env_CASES)
    THEN REWRITE_TAC [IB_Opcode_inE; IB_Addr_inE; IB_Data_inE; IB_BS_inE;
                      IB_BE_inE; IBAM_Opcode_inE; Rst_Opcode_inE]
   );;

%-----------------------------------------------------------------------------
   Abstract data type for the output.
-----------------------------------------------------------------------------%

let rt_out =
   define_type 'rt_out'
               'rt_out = RTOut ibsop wordnn';;

let IB_Opcode_outO = new_recursive_definition
    false
    rt_out
    'IB_Opcode_outO'
    "IB_Opcode_outO (RTOut IB_Opcode_out IB_Data_out)
     = IB_Opcode_out";;

let IB_Data_outO = new_recursive_definition
    false
    rt_out
    'IB_Data_outO'
    "IB_Data_outO (RTOut IB_Opcode_out IB_Data_out)
     = IB_Data_out";;

let Out_CASES =
   prove_cases_thm (prove_induction_thm rt_out);;

let Out_Selectors_Work = prove_thm
   ('Out_Selectors_Work',
```

```
    "!p:rt_out.
     p = (RTOut (IB_Opcode_out0 p) (IB_Data_out0 p))",
    GEN_TAC
    THEN STRUCT_CASES_TAC (SPEC "p:rt_out" Out_CASES)
    THEN REWRITE_TAC [IB_Opcode_out0; IB_Data_out0]
   );;

%-------------------------------------------------------------------------------   Memory Target
Predicates.
-------------------------------------------------------------------------------%

let CBusAddrP = new_definition
   ('CBusAddrP',
    "! (a :wordn) . CBusAddrP a = ELEMENT a (29)"
   );;

let PRegAddrP = new_definition
   ('PRegAddrP',
    "! (a :wordn) .
     PRegAddrP a = ~(ELEMENT a (29)) /\ (SUBARRAY a (23,22) = WORDN 1 3)"
   );;

let LMemAddrP = new_definition
   ('LMemAddrP',
    "! (a :wordn) .
     LMemAddrP a = ~(ELEMENT a (29)) /\ ~(SUBARRAY a (23,22) = WORDN 1 3)"
   );;

let Reg0P = new_definition
   ('Reg0P',
    "! (a :wordn) . Reg0P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 0)"
   );;

let Reg1P = new_definition
   ('Reg1P',
    "! (a :wordn) . Reg1P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 1)"
   );;

let Reg2P = new_definition
   ('Reg2P',
    "! (a :wordn) . Reg2P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 2)"
   );;

let Reg3P = new_definition
   ('Reg3P',
    "! (a :wordn) . Reg3P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 3)"
   );;

let Reg4P = new_definition
   ('Reg4P',
    "! (a :wordn) . Reg4P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 4)"
   );;

let Reg5P = new_definition
   ('Reg5P',
    "! (a :wordn) . Reg5P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 5)"
   );;

let Reg6P = new_definition
   ('Reg6P',
    "! (a :wordn) . Reg6P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 6)"
   );;

let Reg7P = new_definition
   ('Reg7P',
    "! (a :wordn) . Reg7P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 7)"
   );;

let Reg8P = new_definition
   ('Reg8P',
    "! (a :wordn) . Reg8P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 8)"
   );;
```

```
let Reg9P = new_definition
   ('Reg9P',
    "! (a :wordn) . Reg9P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 9)"
   );;

let Reg10P = new_definition
   ('Reg10P',
    "! (a :wordn) . Reg10P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 10)"
   );;

let Reg11P = new_definition
   ('Reg11P',
    "! (a :wordn) . Reg11P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 11)"
   );;

let Reg12P = new_definition
   ('Reg12P',
    "! (a :wordn) . Reg12P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 12)"
   );;

let Reg13P = new_definition
   ('Reg13P',
    "! (a :wordn) . Reg13P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 13)"
   );;

let Reg14P = new_definition
   ('Reg14P',
    "! (a :wordn) . Reg14P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 14)"
   );;

let Reg15P = new_definition
   ('Reg15P',
    "! (a :wordn) . Reg15P a = PRegAddrP a /\ (SUBARRAY a (3,0) = WORDN 3 15)"
   );;

close_theory();;


%-----------------------------------------------------------------------------

   File:      rtransp_def.ml

   Author:    (c) D.A. Fura 1993

   Date:      1 March 1993

   This file contains the ml source for the trans-level specification of the
   R-Port of the FTEP PIU, an ASIC developed by the Embedded Processing
   Laboratory, Boeing High Technology Center.

-----------------------------------------------------------------------------%

set_search_path (search_path() @ ['/home/elvis6/dfura/ftep/piu/hol/rport/';
                                  '/home/elvis6/dfura/ftep/piu/hol/lib/';
                                  '/home/elvis6/dfura/hol/Library/abs_theory/';
                                  '/home/elvis6/dfura/hol/Library/tools/'
                                 ]);;

set_flag ('timing',true);;

system 'rm rtransp_def.th';;

new_theory 'rtransp_def';;

loadf 'abs_theory';;

map new_parent ['rtauxp_def';'array_def';'wordn_def'];;

let REP_ty = abs_type_info (theorem 'piuaux_def' 'REP');;

%-----------------------------------------------------------------------------
```

```
   Next state definition for R-Port instructions.
-------------------------------------------------------------------------------%

let RT_Write_NSF= new_definition
   ('RT_Write_NSF',
    "! (s :rt_state) (e :rt_env) .
     RT_Write_NSF s e =
        let RT_icr = RT_icrS s and
            RT_gcr = RT_gcrS s and
            RT_ccr = RT_ccrS s and
            RT_sr = RT_srS s and
            RT_ctr0_in = RT_ctr0_inS s and
            RT_ctr1_in = RT_ctr1_inS s and
            RT_ctr2_in = RT_ctr2_inS s and
            RT_ctr3_in = RT_ctr3_inS s and
            RT_ctr0 = RT_ctr0S s and
            RT_ctr1 = RT_ctr1S s and
            RT_ctr2 = RT_ctr2S s and
            RT_ctr3 = RT_ctr3S s in
        let IB_Addr_in = IB_Addr_inE e and
            IB_Data_in = IB_Data_inE e and
            IB_BS_in = IB_BS_inE e and
            IB_BE_in = IB_BE_inE e in
        let new_RT_icr =
           (((Reg0P IB_Addr_in) /\ (VAL 1 IB_BS_in = 0)) =>
                 (ANDN 31 (ELEMENT IB_Data_in (0)) RT_icr) |
            ((Reg15P IB_Addr_in) /\ (VAL 1 IB_BS_in = 1)) =>
                 (ANDN 31 (ELEMENT IB_Data_in (1)) RT_icr) |
            ((Reg14P IB_Addr_in) /\ (VAL 1 IB_BS_in = 2)) =>
                 (ANDN 31 (ELEMENT IB_Data_in (2)) RT_icr) |
            ((Reg13P IB_Addr_in) /\ (VAL 1 IB_BS_in = 3)) =>
                 (ANDN 31 (ELEMENT IB_Data_in (3)) RT_icr) |
            (Reg1P IB_Addr_in) =>
               (ORN 31 (ELEMENT IB_Data_in (0)) RT_icr) |
            ((Reg0P IB_Addr_in) /\ (VAL 1 IB_BS_in >= 1)) =>
                 (ANDN 31 (ELEMENT IB_Data_in (1))
                          (ORN 31 (ELEMENT IB_Data_in (0)) RT_icr)) |
            ((Reg15P IB_Addr_in) /\ (VAL 1 IB_BS_in >= 2)) =>
                 (ANDN 31 (ELEMENT IB_Data_in (2))
                          (ORN 31 (ELEMENT IB_Data_in (1)) RT_icr)) |
            ((Reg14P IB_Addr_in) /\ (VAL 1 IB_BS_in >= 3)) =>
                 (ANDN 31 (ELEMENT IB_Data_in (3))
                      (ORN 31 (ELEMENT IB_Data_in (2)) RT_icr)) | RT_icr) in        let new_RT_gcr =
          ((Reg2P IB_Addr_in) => ELEMENT IB_Data_in (0) |
           (Reg1P IB_Addr_in /\
              (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
           (Reg0P IB_Addr_in /\
              (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
           (Reg15P IB_Addr_in /\
              (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_gcr) in
        let new_RT_ccr =
           ((Reg3P IB_Addr_in) => ELEMENT IB_Data_in (0) |
            (Reg2P IB_Addr_in /\
               (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
            (Reg1P IB_Addr_in /\
               (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
            (Reg0P IB_Addr_in /\
               (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ccr) in
        let new_RT_sr = (ARBN:wordn) in
        let new_RT_ctr0_in =
           ((Reg8P IB_Addr_in) => ELEMENT IB_Data_in (0) |
            (Reg7P IB_Addr_in /\
               (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
            (Reg6P IB_Addr_in /\
               (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
            (Reg5P IB_Addr_in /\
               (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr0_in) in
         let new_RT_ctr1_in =
           ((Reg9P IB_Addr_in) => ELEMENT IB_Data_in (0) |
            (Reg8P IB_Addr_in /\
               (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
            (Reg7P IB_Addr_in /\
```

```
            (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
         (Reg6P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr1_in) in
      let new_RT_ctr2_in =
         ((Reg10P IB_Addr_in) => ELEMENT IB_Data_in (0) |
          (Reg9P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
          (Reg8P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
          (Reg7P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr2_in) in
      let new_RT_ctr3_in =
         ((Reg11P IB_Addr_in) => ELEMENT IB_Data_in (0) |
          (Reg10P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
          (Reg9P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
          (Reg8P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr3_in) in
       let new_RT_ctr0 =
         ((Reg12P IB_Addr_in) => ELEMENT IB_Data_in (0) |
          (Reg11P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
          (Reg10P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
          (Reg9P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr0) in
      let new_RT_ctr1 =
         ((Reg13P IB_Addr_in) => ELEMENT IB_Data_in (0) |
          (Reg12P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
          (Reg11P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
          (Reg10P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr1) in
      let new_RT_ctr2 =
         ((Reg14P IB_Addr_in) => ELEMENT IB_Data_in (0) |
          (Reg13P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
          (Reg12P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 2)) =>  ELEMENT IB_Data_in (2) |
          (Reg11P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr2) in
       let new_RT_ctr3 =
         ((Reg15P IB_Addr_in) => ELEMENT IB_Data_in (0) |
          (Reg14P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 1)) => ELEMENT IB_Data_in (1) |
          (Reg13P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 2)) => ELEMENT IB_Data_in (2) |
          (Reg12P IB_Addr_in /\
            (VAL 1 IB_BS_in >= 3)) => ELEMENT IB_Data_in (3) | RT_ctr3) in

      (RTState new_RT_icr new_RT_gcr new_RT_ccr new_RT_sr new_RT_ctr0_in
               new_RT_ctr1_in new_RT_ctr2_in new_RT_ctr3_in new_RT_ctr0
               new_RT_ctr1 new_RT_ctr2 new_RT_ctr3)"
   );;

let RT_Read_NSF = new_definition
   ('RT_Read_NSF',
    "! (s :rt_state) (e :rt_env) .
     RT_Read_NSF s e =
        let new_RT_icr = RT_icrS s in
        let new_RT_gcr = RT_gcrS s in
        let new_RT_ccr = RT_ccrS s in
        let new_RT_sr = RT_srS s in
        let new_RT_ctr0_in = RT_ctr0_inS s in
        let new_RT_ctr1_in = RT_ctr1_inS s in
        let new_RT_ctr2_in = RT_ctr2_inS s in
        let new_RT_ctr3_in = RT_ctr3_inS s in
        let new_RT_ctr0 = RT_ctr0S s in
        let new_RT_ctr1 = RT_ctr1S s in
```

```
        let new_RT_ctr2 = RT_ctr2S s in
        let new_RT_ctr3 = RT_ctr3S s in

        (RTState new_RT_icr new_RT_gcr new_RT_ccr new_RT_sr new_RT_ctr0_in
                 new_RT_ctr1_in new_RT_ctr2_in new_RT_ctr3_in new_RT_ctr0
                 new_RT_ctr1 new_RT_ctr2 new_RT_ctr3)"
   );;

let RT_Idle_NSF = new_definition
   ('RT_Idle_NSF',
    "! (s :rt_state) (e :rt_env) .
     RT_Idle_NSF s e =
        let new_RT_icr = RT_icrS s in
        let new_RT_gcr = RT_gcrS s in
        let new_RT_ccr = RT_ccrS s in
        let new_RT_sr = RT_srS s in
        let new_RT_ctr0_in = RT_ctr0_inS s in
        let new_RT_ctr1_in = RT_ctr1_inS s in
        let new_RT_ctr2_in = RT_ctr2_inS s in
        let new_RT_ctr3_in = RT_ctr3_inS s in
        let new_RT_ctr0 = RT_ctr0S s in
        let new_RT_ctr1 = RT_ctr1S s in
        let new_RT_ctr2 = RT_ctr2S s in
        let new_RT_ctr3 = RT_ctr3S s in

        (RTState new_RT_icr new_RT_gcr new_RT_ccr new_RT_sr new_RT_ctr0_in
                 new_RT_ctr1_in new_RT_ctr2_in new_RT_ctr3_in new_RT_ctr0
                 new_RT_ctr1 new_RT_ctr2 new_RT_ctr3)"
   );;

%-----------------------------------------------------------------------------
   Output definition for R-Port instructions.
-----------------------------------------------------------------------------%

let RT_Write_OF = new_definition
   ('RT_Write_OF',
    "! (s :rt_state) (e :rt_env) .
     RT_Write_OF s e =
        let IB_Opcode_out = IBS_Ready in
        let IB_Data_out = (ARBN:num->wordn) in

        (RTOut IB_Opcode_out IB_Data_out)"
   );;

let RT_Read_OF = new_definition
   ('RT_Read_OF',
    "! (s :rt_state) (e :rt_env) .
     RT_Read_OF s e =
        let IB_Opcode_out = IBS_Ready in
        let bs = VAL 1 (IB_BS_inE e) in
        let d0 = (Reg0P (IB_Addr_inE e)) => RT_icrS s |
                 (Reg1P (IB_Addr_inE e)) => RT_icrS s |
                 (Reg2P (IB_Addr_inE e)) => RT_gcrS s |
                 (Reg3P (IB_Addr_inE e)) => RT_ccrS s |
                 (Reg4P (IB_Addr_inE e)) => RT_srS s |
                 (Reg8P (IB_Addr_inE e)) => RT_ctr0_inS s |
                 (Reg9P (IB_Addr_inE e)) => RT_ctr1_inS s |
                 (Reg10P (IB_Addr_inE e)) => RT_ctr2_inS s |
                 (Reg11P (IB_Addr_inE e)) => RT_ctr3_inS s |
                 (Reg12P (IB_Addr_inE e)) => RT_ctr0S s |
                 (Reg13P (IB_Addr_inE e)) => RT_ctr1S s |
                 (Reg14P (IB_Addr_inE e)) => RT_ctr2S s |
                 (Reg15P (IB_Addr_inE e)) => RT_ctr3S s | ARBN in
        let d1 = (bs > 0)
                   => (Reg15P (IB_Addr_inE e)) => RT_icrS s |
                      (Reg0P (IB_Addr_inE e)) => RT_icrS s |
                      (Reg1P (IB_Addr_inE e)) => RT_gcrS s |
                      (Reg2P (IB_Addr_inE e)) => RT_ccrS s |
                      (Reg3P (IB_Addr_inE e)) => RT_srS s |
                      (Reg7P (IB_Addr_inE e)) => RT_ctr0_inS s |
                      (Reg8P (IB_Addr_inE e)) => RT_ctr1_inS s |
                      (Reg9P (IB_Addr_inE e)) => RT_ctr2_inS s |
```

```
                    (Reg10P (IB_Addr_inE e)) => RT_ctr3_inS s |
                    (Reg11P (IB_Addr_inE e)) => RT_ctr0S s |
                    (Reg12P (IB_Addr_inE e)) => RT_ctr1S s |
                    (Reg13P (IB_Addr_inE e)) => RT_ctr2S s |
                    (Reg14P (IB_Addr_inE e)) => RT_ctr3S s | ARBN
                | ARBN  in
        let d2 = (bs > 1)
                => (Reg14P (IB_Addr_inE e)) => RT_icrS s |
                    (Reg15P (IB_Addr_inE e)) => RT_icrS s |
                    (Reg0P (IB_Addr_inE e)) => RT_gcrS s |
                    (Reg1P (IB_Addr_inE e)) => RT_ccrS s |
                    (Reg2P (IB_Addr_inE e)) => RT_srS s |
                    (Reg6P (IB_Addr_inE e)) => RT_ctr0_inS s |
                    (Reg7P (IB_Addr_inE e)) => RT_ctr1_inS s |
                    (Reg8P (IB_Addr_inE e)) => RT_ctr2_inS s |
                    (Reg9P (IB_Addr_inE e)) => RT_ctr3_inS s |
                    (Reg10P (IB_Addr_inE e)) => RT_ctr0S s |
                    (Reg11P (IB_Addr_inE e)) => RT_ctr1S s |
                    (Reg12P (IB_Addr_inE e)) => RT_ctr2S s |
                    (Reg13P (IB_Addr_inE e)) => RT_ctr3S s | ARBN
                | ARBN  in
        let d3 = (bs > 2)
                => (Reg13P (IB_Addr_inE e)) => RT_icrS s |
                    (Reg14P (IB_Addr_inE e)) => RT_icrS s |
                    (Reg15P (IB_Addr_inE e)) => RT_gcrS s |
                    (Reg0P (IB_Addr_inE e)) => RT_ccrS s |
                    (Reg1P (IB_Addr_inE e)) => RT_srS s |
                    (Reg5P (IB_Addr_inE e)) => RT_ctr0_inS s |
                    (Reg6P (IB_Addr_inE e)) => RT_ctr1_inS s |
                    (Reg7P (IB_Addr_inE e)) => RT_ctr2_inS s |
                    (Reg8P (IB_Addr_inE e)) => RT_ctr3_inS s |
                    (Reg9P (IB_Addr_inE e)) => RT_ctr0S s |
                    (Reg10P (IB_Addr_inE e)) => RT_ctr1S s |
                    (Reg11P (IB_Addr_inE e)) => RT_ctr2S s |
                    (Reg12P (IB_Addr_inE e)) => RT_ctr3S s | ARBN
                | ARBN  in
        let IB_Data_out =
              ALTER (ALTER (ALTER (ALTER ARBN(0) d0)(1) d1)(2) d2)(3) d3 in


        (RTOut IB_Opcode_out IB_Data_out)"
   );;

let RT_Idle_OF = new_definition
   ('RT_Idle_OF',
    "! (s :rt_state) (e :rt_env) .
     RT_Idle_OF s e =
        let IB_Opcode_out = IBS_Idle in
        let IB_Data_out = (ARBN:num->wordn) in

        (RTOut IB_Opcode_out IB_Data_out)"
   );;

%----------------------------------------------------------------------------
   R-Port interpreter definition.
----------------------------------------------------------------------------%

let RT_Exec = new_definition
   ('RT_Exec',
    "! (rti :RTI) (s :timeT->rt_state) (e :timeT->rt_env) (p :timeT->rt_out)
       (t :timeT) .
     RT_Exec rti s e p t =
        (IBAM_Opcode_inE (e t) = IBAM_ProcP) /\
        ((rti = RT_Write) => (IB_Opcode_inE (e t) = PBM_WritePIU) |
         (rti = RT_Read) =>  (IB_Opcode_inE (e t) = PBM_ReadPIU)
       % (rti = RT_Idle) % | ((IB_Opcode_inE (e t) = PBM_WriteLM) \/
                              (IB_Opcode_inE (e t) = PBM_ReadLM) \/
                              (IB_Opcode_inE (e t) = PBM_WriteCB) \/
                              (IB_Opcode_inE (e t) = PBM_ReadCB)))"
   );;

let RT_PreC = new_prim_rec_definition
```

```
   ('RT_PreC',
    "(RT_PreC (rti:RTI) s e p 0 = T) /\
     (RT_PreC rti s e p (SUC t) =
        (RT_Exec RT_Write s e p t /\ RT_PreC RT_Write s e p t) \/
        (RT_Exec RT_Read s e p t /\ RT_PreC RT_Read s e p t) \/
        (RT_Exec RT_Idle s e p t /\ RT_PreC RT_Idle s e p t))"
   );;

let RT_PostC = new_definition
   ('RT_PostC',
    "! (rti :RTI) (s :timeT->rt_state) (e :timeT->rt_env)
       (p :timeT->rt_out) (t :timeT) .
     RT_PostC rti s e p t =
        (rti = RT_Write) => ((s (t+1) = RT_Write_NSF (s t) (e t)) /\
                            (p t = RT_Write_OF (s t) (e t))) |
        (rti = RT_Read) => ((s (t+1) = RT_Read_NSF (s t) (e t)) /\
                           (p t = RT_Read_OF (s t) (e t)))
      % (rti = RT_Idle) % | ((s (t+1) = RT_Idle_NSF (s t) (e t)) /\
                            (p t = RT_Idle_OF (s t) (e t)))"
   );;

let RT_Correct = new_definition
   ('RT_Correct',
    "! (rti :RTI) (s :timeT->rt_state) (e :timeT->rt_env)
       (p :timeT->rt_out) (t :timeT) .
     RT_Correct rti s e p t =
        RT_Exec rti s e p t /\
        RT_PreC rti s e p t
        ==>
        RT_PostC rti s e p t"
   );;

let RTSet_Correct = new_definition
   ('RTSet_Correct',
    "! (s :timeT->rt_state) (e :timeT->rt_env) (p :timeT->rt_out) .
     RTSet_Correct s e p = !(rti:RTI)(t:timeT). RT_Correct rti s e p t"
   );;

close_theory();;
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November, 1993	Contractor Report

4. TITLE AND SUBTITLE
Towards the Formal Specification of the Requirements and Design of a Processor Interface Unit--HOL Listings

5. FUNDING NUMBERS
C NAS1-18586

WU 505-64-10-07

6. AUTHOR(S)
David A. Fura, Phillip J. Windley*, and Gerald C. Cohen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Boeing Defense & Space Group
P.O. Box 3707, M/S 4C-70
Seattle, WA 98124-2207

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
NASA Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CR-191465

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Sally C. Johnson
Task 10 Report
*University of Idaho, Moscow, ID

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 62

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report describes work to formally specify the requirements and design of a Processor Interface Unit (PIU), a single-chip subsystem providing memory interface, bus interface, and additional support services for a commercial microprocessor within a fault-tolerant computer system. This system, the Fault-Tolerant Embedded Processor (FTEP), is targeted towards applications in avionics and space requiring extremely high levels of mission reliability, extended maintenance-free operation, or both. This report contains the Higher Order Logic (HOL) listings of specifications of a nontrivial subset of the PIU at several levels of abstracting from gate-level models of individual components to a transaction level; specification of the PIU's behavior. The approaches used in creating these specifications are documented in a companion NASA contractor report entitled "Towards the Formal Specification of the Requirements and Design of a Processor Interface Unit."

14. SUBJECT TERMS	15. NUMBER OF PAGES
Formal methods; Formal specification; Formal verification; Fault tolerance; Reliability; Specification	249
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		